Scholar Commons
USF University of South Florida

University of South Florida
Scholar Commons

Graduate Theses and Dissertations | Graduate School

January 2012

Human Motion Tracking for Assisting Balance Training and Control of a Humanoid Robot

Ahmad Adli Manasrah
University of South Florida, manasrah@mail.usf.edu

Follow this and additional works at: http://scholarcommons.usf.edu/etd


Part of the American Studies Commons, Mechanical Engineering Commons, and the Medicine and Health Sciences Commons

Scholar Commons Citation
Manasrah, Ahmad Adli, "Human Motion Tracking for Assisting Balance Training and Control of a Humanoid Robot" (2012). *Graduate Theses and Dissertations.*
http://scholarcommons.usf.edu/etd/4141

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.

المنارة للاستشارات

www.manaraa.com

Human Motion Tracking for Assisting Balance Training and Control of a Humanoid Robot

by

Ahmad Manasrah

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Mechanical Egineering
Department of Mechanical Engineering
College of Engineering
University of South Florida

Co-Major Professor: Kyle Reed, Ph.D.
Co-Major Professor: Rasim Guldiken, Ph.D.
Craig Lusk, Ph.D

Date of Approval:
June 22, 2012

Keywords: Kinect, Center of Mass, Depth,
Skeleton, Visual Feedback

Copyright © 2012, Ahmad Manasrah

www.manaraa.com

Acknowledgments

I wish to acknowledge my co-major professors Dr. Kyle Reed and Dr. Rasim Guldiken for their help and support over the past two years. I wish to thank the other committee member Dr. Craig Lusk for the valuable information and advice. I also would like to acknowledge all the faculty members in the Mechanical Engineering Department for their support during my study.

www.manaraa.com

Table of Contents

www.manaraa.com

www.manaraa.com

List of Tables

www.manaraa.com

List of Figures

www.manaraa.com

www.manaraa.com

Abstract

This project illustrates the use of the human's ability to balance according to his center of gravity as demonstrated in two applications. The center of gravity of a human is explained in detail in order to use it in controlling the Aldebaran NAO robot and in the robot-assisted balance training.

The first application explains how a humanoid robot can mimic a human's movements via a three dimensional depth sensor where the sensor analyzes the position of a user's limbs and how the robot can lift one foot and balance on the other by redistributing the its body mass when the user lifts his foot. The results showed that this algorithm enabled NAO to successfully mimic the users' arms, and was able to balance on one foot by repositioning its center of mass.

The second application investigates how individuals with stroke lean when undergoing robot-assisted balance training. In some instances, they can develop inappropriate leaning behaviors during the training. The Kinect sensor is used to assist in optimizing patients' results by integrating it with the training program. The results showed that the Kinect sensor can improve the efficiency of the process by giving users graphical information about their mass distribution and whether they are leaning correctly or not.

www.manaraa.com

Chapter 1: Introduction

Since the Microsoft Kinect sensor has been released in 2009, many developers have built applications and projects that use the sensor's ability to sense 3D images and detect human body presence. Some of the projects focused on using the body gestures to control a specific application through an interface. The video games industry is interested in particular because of its revolutionary feature where players can control video games using their body gestures. Besides this remarkable breakthrough in video games, Kinect was being used in many applications such as: educational programs, security applications, and physical rehabilitation.

All these features and abilities have made Kinect an interesting field of study, thus, it was decided to test the sensor in real-life applications to investigate how efficient Kinect could be. The objectives of this thesis are:

- Studying the human's ability to balance according to his center of gravity in two applications.
- Using the Kinect sensor features to calculate the center of mass of a human body.
- Applying human motion tracking with a humanoid robot by using the Kinect sensor as an interface device between the user (controller) and the robot.
- Studying a physical rehabilitation process, understanding the procedures and improving its results.

www.manaraa.com

Two different applications were chosen to study the human's balance and test the Kinect sensor. The first application was using Kinect as an interface between users and the Aldebaran NAO robot, which is a programmable humanoid robot that has remarkable features. The objective of this interface is to enable NAO to imitate the human's arm and leg movements. The Kinect sensor will create a set of data about a human's body joints, and these data will be used on NAO to mimic the moves of the human arms. Having NAO recognize the human movements is the first step of creating a hands-free controllable robot that can do many tasks wirelessly. The Kinect sensor may be considered the future of controlling humanoid robotsThe second application was using the Kinect sensor in the robot-assisted balance training (RABT) which is a rehabilitation program that was developed and built by Dr. Seok Hun Kim, PhD, and Dr. Kyle Reed, PhD, at the University of South Florida. Physical therapy and rehabilitation can be very slow and expensive especially for stroke patients who suffer from walking disabilities and balance problems. The RABT trains patients to adapt their balance by applying an external pulling force to their waists [14]. This method introduces a real-time visual feedback to patients that has shown significantly improved results. The purpose of using the Kinect sensor with RABT is to enhance the visual feedback, and improve patients' ability to balance during the training.

Both applications are explained in detail in the following chapters. Chapter two introduces the Kinect sensor, discusses its working principle, and highlights the sensor's advantages and limitations. It also explains the principles of calculating the center of mass of the human body, and how we can use these calculations with the data of the Kinect sensor.

www.manaraa.com

Chapter three introduces the Aldebaran NAO and shows a brief background of the robot. It also analyses its body extremities and compares between them and the human extremities. The chapter shows how these information are used with the Kinect sensor to have NAO react with human arm movements by calculating the arm angles. The chapter also discusses how NAO can balance on one foot by redistributing its body mass as controlled by a person lifting his foot. And finally, chapter three provides the results and conclusions of using the Kinect sensor to control NAO at the end of the chapter.

Chapter four starts with a background about the robot-assisted balance training, and explains how the training is done and what are the expected results. The chapter discusses the common mistakes that patients usually do during the training where they shift their body weight improperly, and the effects of these mistakes on the results. The chapter connects these inforamtion with the Kinect sensor data to help patients undergoing RABT to improve their training by giving them visual feedback about their leaning pattern. Finally, chapter four provides the results of integrating the Kinect sensor with the robot-assisted balance training.

Chapter five introduces the future works that may expand the use of the Kinect sensor with these two applications.

www.manaraa.com

Chapter 2: About Microsoft Kinect

2.1 Introduction

Infrared distance measurement sensors have been widely used in various applications and projects to measure the distance of objects, and they are commonly used in mobile robotics to avoid obstacles and sense the presence of objects. These sensors consist of transmitters and receivers where the sensor transmits an infrared beam to an object and receives the reflection of the infrared light from the object's surface. The infrared sensor is an example of determining the distance of objects, but in order to create a depth image of the surrounding environment, an array of infrared sensors are needed as a simple idea to create a 3D image. For instance, Panasonic has developed a 3D image sensor called "D-imager" that uses an array of infrared LED's to create a three dimensional image and a time-of-flight technology where the time required for the light to return to the receiver after being reflected from an object is calculated by a processor. D-imager's operating range is 1.2 m ~ 9 m, and works at frame rates up to 30 fps [1].

Microsoft Kinect is a sensing device owned by Microsoft, originally designed for Xbox 360 video game console which enables players to control their video games using their own body gestures and voice commands. The device was announced in 2009 under the name of "Project Natal", and has a unique 3D sensing technology which was developed by the Israeli company PrimeSense [2].

www.manaraa.com

2.2 Kinect Overview

The Kinect sensor consists of an RGB camera, a 3D depth sensor in the front of the Kinect, and multiple microphone arrays at the sides. The sensor also has a motorized tilt that allows users to change the viewing angle plus or minus 27 degrees. The depth sensor consists of two parts, an infrared projector, which is located on the left side of the RGB camera, and receiver which is located on the right side of the camera. Figure (2-1) shows the components of the sensor.

Figure 2-1: Microsoft Kinect Components [3].

2.3 Working Principle

The Kinect sensor provides 3D depth information of any object in front of it, which allows users to use simpler ways to detect these objects, while most of the other systems only provide 2D information, which requires more algorithms to detect the same objects. Figure (2-2) shows the internal components of the Kinect sensor. The most important components are the infrared transmitter and the receiver which are shown in the light green color. The internal algorithm sends and receives data from the depth sensor to process them.

www.manaraa.com

The most common way to calculate the distance of an object is by projecting infrared light on it and then calculating the time it takes the light to reflect off the object and return to the receiver, but the Kinect sensor works in a different way; the infrared projector projects a fixed pattern of infrared spots onto the object, then the receiver captures a shifted grid of these spots, the processor then calculates the offset of each of the spots to generate a depth map. The Kinect sensor can measure the distance of an object 2 meters away within 1 cm accuracy [2].

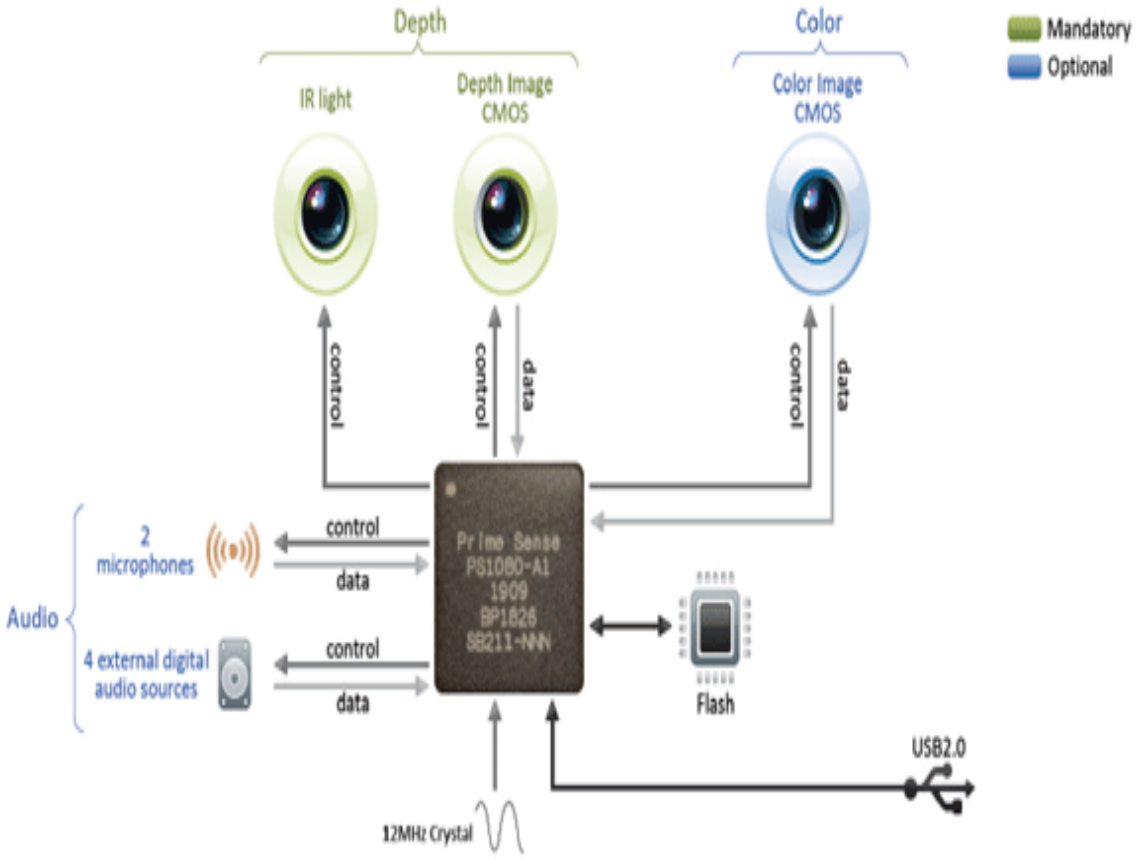


Figure 2-2: The Internal Components of Kinect [2].

www.manaraa.com

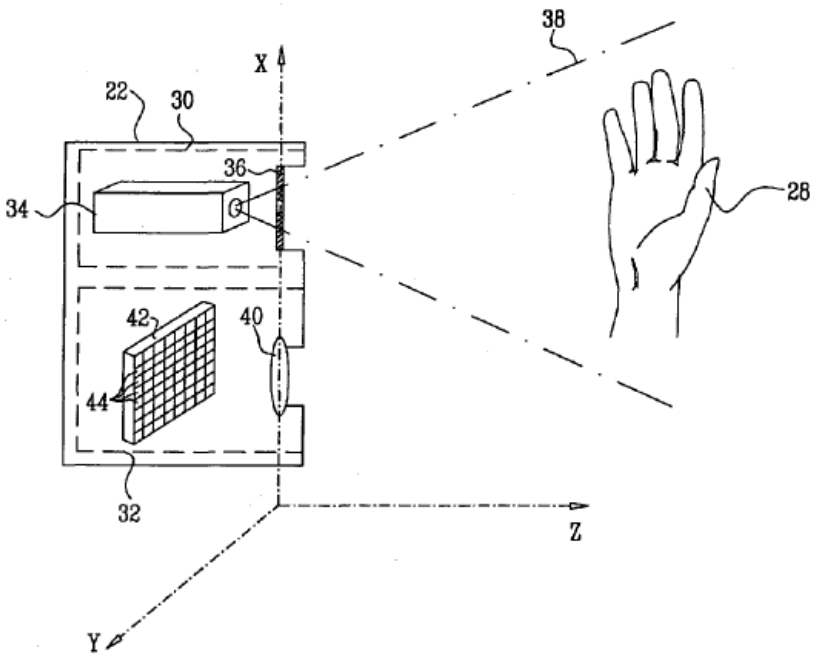


Figure 2-3: A Simple Representation of How Kinect Works [4].

Figure (2-3) explains how Kinect works, it represents the "depth mapping using projected patterns" [4]. An illumination assembly (30) which has a single transparency (36) with a fixed pattern of spots. This assembly has a light source (34) which illuminates through the fixed pattern transparency, the pattern then reflects from the object (28) to the capture assembly (32) where the image of the pattern is focused by a lens (40), then captured by an image sensor (42). After that, the reflected image is processed to build a 3D map of the object [4].

2.4 Software

When the Kinect sensor was released by Microsoft, the device was only dedicated to video games. After that, PrimeSense introduced the first open source software for the sensor (OpenNI) which enabled developers to access the main features of the sensor

www.manaraa.com

using Windows, and Linux machines. The raw data that was accessed allowed developers to start their own projects and use the 3D depth feature. In 2011, a group of researchers have done a real-time 3D reconstruction to the surrounding environment using the depth data from the Kinect sensor [17].

2.5 Skeleton Tracking

Microsoft developed a software of skeleton tracking for the Kinect sensor that enables Xbox players to use their body movements to control video games. This software was only accessed by Microsoft, and that only alternative software that supports this feature was NITE from PrimeSense. In June, 2011 Microsoft announced the first official release of Kinect SDK (Software Development Kit) and it opened the door for more developing with depth and skeleton tracking information. Skeletal tracking works only when the depth data is enabled.

Building a skeleton requires a depth image of a human body. The sensor's algorithm designs an intermediate body parts representation to map the body [5], some of these parts are defined as body joints, and some of them represent the links that connect the joints together. These parts are color coded, and the algorithm recognizes these colors to detect left and right sides of the body then creates the body joints depending on these color coded parts. Figure (2-4) explains how the skeleton is detected. The depth image is represented as the gray scale image, and the algorithm detects the human presence in this image and identifies the body parts, after that, the algorithm detects the joints of the human and represents them in 3D space.

www.manaraa.com

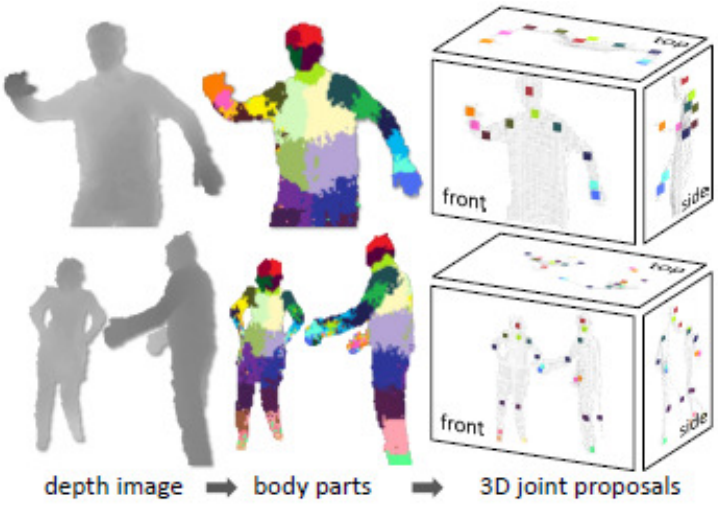


Figure 2-4: A Depth Image of a Human Body [5]. © 2011 IEEE

The algorithm identifies the features of a certain object by determining the consistency of its pixels using this equation [6] at any given pixel (x):

$$f_\theta(I,x) = d_I\left(x + \frac{u}{d_I(x)}\right) - d_I\left(x + \frac{v}{d_I(x)}\right) \qquad \text{equation (1)}$$

where:

- $d_I(x)$: the depth at pixel x.
- u and v : the offsets of the pixel x, at image I, where I is the image number.

The offsets are divided by $d_I(x)$, and added to x. If the offset pixel lies on the background, it will have a large depth value (since it is farther than x), which means that the offset value will also be large. Thus, this offset pixel does not belong to the same surface area where x is taken from. When the offset pixel's value is close to zero, it means that both pixels belong to the same object surface, or in other words, to the same body part.

www.manaraa.com

Figure (2-5) explains the two cases where the sensor is detecting which part of the body the pixel x belongs to.

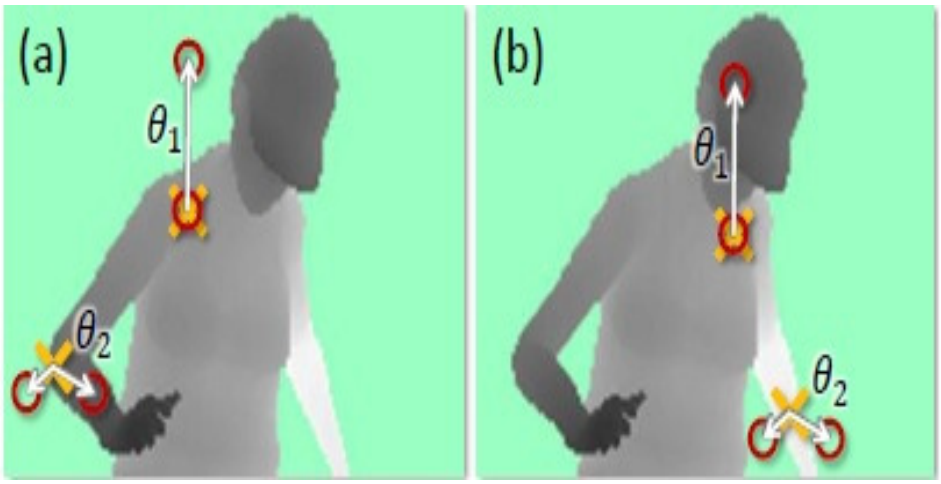


Figure 2-5: Detecting the Body Parts. (a) Shows a large pixel offset. (b) Shows a small pixel offset [5]. © 2011 IEEE

The yellow cross represents the pixel x, and the red circles represent the offsets of x in two directions (u and v). In figure (2-5 - a) the offset pixel of the function $f_{\theta 1}$ is too large, while it is too small in figure (2-5 - b). The algorithm uses this information to construct a line that connects two pixels from two different part areas to create a part of the skeleton like the spine or the arm. This method depends only on a mathematical operation and does not need any kind of image processing, and its results can be used directly to create a skeleton.

The algorithm is trained on thousands of randomly different training images to increase the prediction accuracy of the skeleton.

www.manaraa.com

2.6 Advantages and Limitations

The Microsoft Kinect sensor has many advantages that differentiate it from other distance measurement sensors, such as:

- Voice and facial recognition, Kinect sensor has the ability to receive voice commands in different languages to do certain operations like adjusting the tilt motor for instance. It also supports facial recognition and the face tracking capability which is now supported in the new version of SDK (Kinect SDK v-1.5).
- The skeleton tracking feature, where a person's skeleton can be constructed by detecting his/her body joints.
- Easy installation, the Kinect sensor has a software toolkit that can be downloaded from Microsoft and it does not need any data input devices or special connections, the Kinect sensor comes with a power supply cable and a USB.
- The price of Kinect sensor is about $150 which is fairly inexpensive compared to other depth sensors.
- The Kinect sensor has a tilt motor that allows the operator to reposition it at any vertical angle between 27 and -27 degrees.
- The range of detection of the sensor is very good for sighting human bodies from 0.8 m to 4 m.
- The Kinect sensor can work in the dark without any source of light.
- Kinect has a feature of detecting the position of unseen joints, for instance, if a person is standing in front of the Kinect sensor and one of his arms is out of

www.manaraa.com

the sensor's sight range, Kinect can detect the positions of the unseen joints based on the configuration of the other joints.

The Kinect sensor also has some disadvantages that could limit its abilities such as:

- The facial recognition. In some applications, the facial recognition might be a limitation in the working field. the Kinect sensor cannot be used to detect a human skeleton without detecting his/her face first. In this thesis, the Kinect sensor was not very effective by placing it behind a person.
- Working with the Kinect outdoor was not very effective, since the sensor is sensitive to direct light, and may receive different IR light from any source.
- The software of the Kinect sensor does not support skeleton detection with two or more Kinects working at the same time on one machine yet. The depth feature in the software supports up to six Kinect sensors at the same time, but the infrared signals may interfere with each other.
- Despite the Kinect's ability to detect a human presence by recognizing his face and body posture, sometimes it might create a skeleton on some other objects like a table or a chair, especially when the object appears to have parts that may look like extremities.
- The developing in Kinect sensor is mainly directed towards video gaming and entertaining applications.
- The facial detection property may raise some security and privacy questions.

2.7 Other Applications of Kinect

One of the interesting researches that was done with the Kinect sensor was the "3D indoor exploration with a computationally constrained micro-aerial vehicle" [18] that

www.manaraa.com

presents the ability of the Kinect sensor to explore an indoor environment and construct a 3D visualization of it. The Kinect sensor is mounted on a micro-aerial vehicle which flies in the environment. The Kinect sensor collects data of the geometry of the environment and construct a 3D image of it.

Another application that uses the Kinect sensor technology is detecting early stages of serious illness in older adults, and protecting them from falling [19]. Many seniors may not notice the slight change in their activities which may lead to serious injury. Multiple Kinect sensors are used to monitor seniors' activities inside their homes to construct a 3D presentation of the person to read his/her body movements. The sensors detect the walking pattern and the geometry of seniors and send data to health care providers to alert them in case of emergencies.

2.8 The Center of Mass

2.8.1 The Center of Mass of a Human Body

The center of mass of a human in a uniform gravitational field (also known as the center of gravity) is the point where all the masses of the human body can be considered to be concentrated for some purposes, it is located at the center of the torso of the body where the human is standing upright, arms down, "at about 55% of the total height" [8]. The location of the center of mass depends on the pose of the body, for example, if a person is standing up, rising both arms over his/her shoulders, then the center of mass would be roughly over the belly button at about the height of the stomach. The center of mass can even be outside the body, if the person manages to bend his/her upper body to any side.

www.manaraa.com

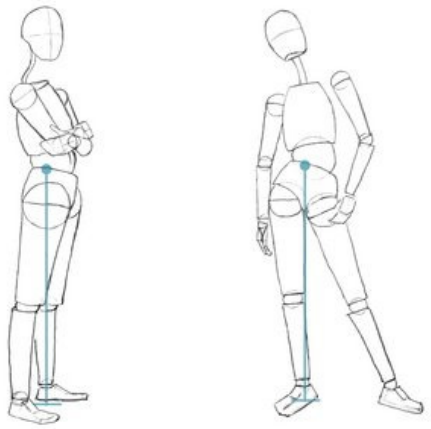


Figure 2-6: Two Manikins with Different Poses [20].

Figure (2-6) shows two manikins with different poses, the one on the left represents a human body standing up, its center of mass is located in the torso pointing down right between the feet, the manikin is balanced because both legs support the center of mass equally. The manikin on the right poses where the body weight is shifted to its right leg, the center of mass is exactly located above the right leg to keep the balance. The manikin's left leg can actually be lifted and still be stable; because the center of mass is over the right leg.

Figure (2-7) shows another two manikins where their center of masses are located outside their bodies. The manikin on the left is bending down in an almost 90 degree angle, the center of mass is located underneath the stomach area outside the body because of the body's mass distribution in this position. The manikin on the right is bending backwards, arms up, the center of mass is also located outside the body.

www.manaraa.com

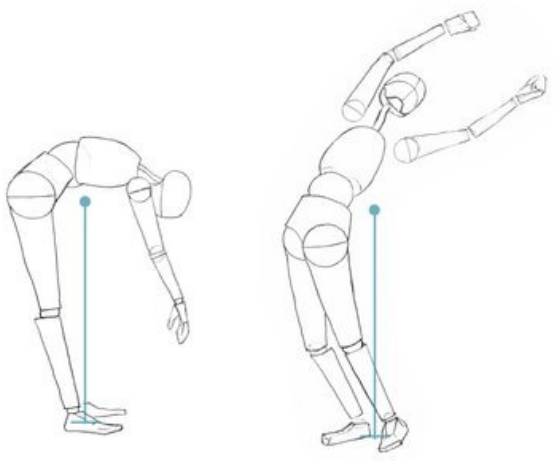


Figure 2-7: The Center of Mass Outside the Body [20].

The human body can maintain its quasi static balance in any standing position as long as the center of mass is located above the support area, which is the area between the feet. The four manikins in figure (2-6) and figure (2-7) are all in different standing positions, but all the center of masses are located above the support areas to keep the balance. If the center of mass shifts outside this area at any standing position in any direction, the body will not maintain balance and start to fall. When the body is moving, the center of mass may not necessarily be located above the support area; walking can be thought of as a state of constantly falling forward where one foot is placed to recover stability and the falling again.

2.8.2 Determining the Center of Mass

We can calculate the center of mass of a single object where its shape and orientation do not change, and because the object's mass will not shift to any direction, the location of the mass center will not change too. The center of mass of a discrete system is at the position $r_{cm}^{\rightarrow}$ [9]:

www.manaraa.com

$$r_{cm}^{\rightarrow} = \sum \frac{r_i^{\rightarrow} m_i}{m_{tot}}$$ Equation (2)

where m_{tot} is the summation of masses for discrete system. Figure (2-8) shows a system of two discrete masses m_1 and m_2 where $m_2 > m_1$. It is obvious that the center of mass of such a system would be between the two masses and closer to the larger one.

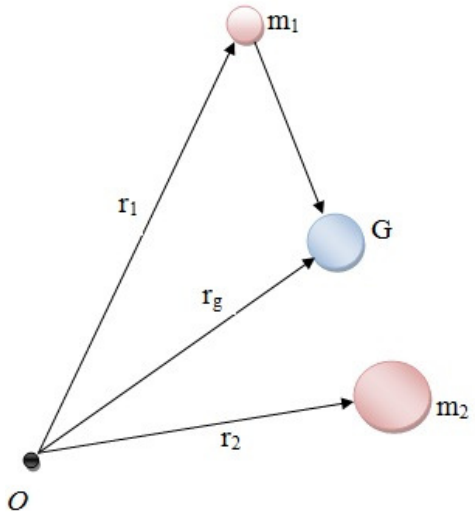


Figure 2-8: The Center of Mass of a Two Point System.

r_1 and r_2 are two vectors that represent how far the two masses are away from the origin point (reference point). According to the equation above, the position of the center of mass of the system $r_G^{\rightarrow}$ is:

$$r_G^{\rightarrow} = \sum \frac{r_i^{\rightarrow} m_i}{m_{tot}}$$

$$= \frac{r_1^{\rightarrow} m_1 + r_2^{\rightarrow} m_2}{m_1 + m_2}$$

www.manaraa.com

$$= \frac{\vec{r_1}(m_1 + m_2) - \vec{r_1}m_2 + \vec{r_2}m_2}{m_1 + m_2}$$

$$= \vec{r_1} + \left(\frac{m_2}{m_1 + m_2}\right)(\vec{r_2} - \vec{r_1}) \qquad \text{Equation (3)}$$

From equation (3), if $m_2 >> m_1$ so that m_1 is negligible, the center of mass will be over m_2. If $m_1 >> m_2$ so that m_2 is negligible, then the center of mass will be over m_1, which makes sense according to equation (2). If the two masses are equal $m_1 = m_2$, the center of mass will be exactly in the middle.

The human body consists of parts connected to each other with joints, and every time the positions of these parts and joints are changed, the distribution of the mass changes depending on the body posture, hence, the location of the center of mass changes. Physicians and scientist tried to calculate the center of mass of a human body with different ways like the "immersion method" where they submerged body segments in water to see how much water was displaced [10]. Other computational methods were used by comparing body segments with cylindrical shapes and circular cones.

Table (2-1) below represents the center of mass of each body segment according to professor Rudolfs Dilliris and Renato Contini's method of "reaction change", the method calculates the reaction force of a board while the subject lies on it. The board is supported by a fixed base at one end and a sensitive weighing scale at the other end. Results showed that the human body can be divided to segments, each segment has a proximal and distal endpoints which represent the human joints. The fractional body mass shows the weight percentage of each segment in the body. For instance, the fractional body mass of the head and trunk is about 53% of the body mass.

www.manaraa.com

Table 2-1: Anthropomorphic Data of Body Segments [10].

Body Segment	Proximal Endpoint	Distal Endpoint	Fractional Body Mass	COM location from Proximal Endpoint
Foot	Ankle Joint Center	Virtual Toe	0.019	0.429
Shank	Knee Joint Center	Ankle Joint Center	0.044	0.433
Thigh	Hip Joint Center	Knee Joint Center	0.115	0.433
Hand & Forearm	Elbow Marker	Wrist Marker	0.025	0.682
Upper Arm	Shoulder Marker	Elbow Marker	0.031	0.436
Head & Trunk	Midpoint between hip joint centers	Midpoint between shoulder markers	0.532	0.54

The fractional body mass varies from person to person depending on the body proportions, it also differs from males to females. The head and trunk segment takes the largest fractional body mass, so the center of mass will always be located close to it. Each segment has its own center of mass where its location depends on the body segment's shape and weight. For example, the center of mass of the thigh is located at 0.433 the distance between the hip joint and the knee joint, it means that the center of mass is closer to the hip joint based on the weight distribution of the thigh.

Figure (2-9) shows the weight distributions of a human body, the sizes of the spheres are proportional to segment masses. This set of masses is similar to the one shown in figure (2-8), and with the same calculations, the center of mass of the human body can be found and its location will always be closer to the biggest mass in the system.

www.manaraa.com

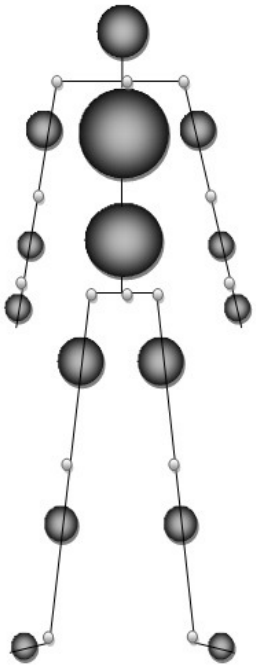


Figure 2-9: Body Mass Distribution.

2.8.3 The Center of Mass and Kinect

Chapter two introduced the Kinect sensor and explained the working principle of the depth measurement and the skeleton feature of the sensor. It also discussed the center of mass of the human body by showing its importance for the human's balance and explaining how to calculate it in a system of masses. All these background information can be used to achieve the objectives of this section, which are:

- To use the ability of the Kinect sensor to construct and track the user's skeleton.
- To calculate the center of mass of the human body using the Kinect sensor as a data input device.

Microsoft Kinect provides depth and location data to each joint in the detected skeleton. The sensor detects 20 point for each skeleton which are located as in figure (2-10) that

www.manaraa.com

represents the famous drawing by Leonardo da Vinci, The Vitruvian man with Kinect's joints located on it.

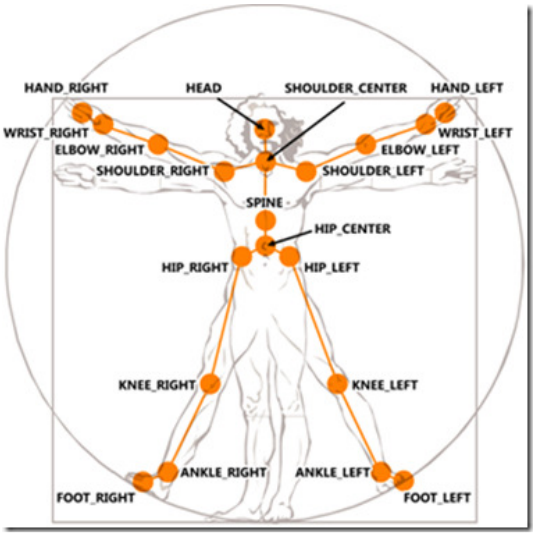


Figure 2-10: The Vitruvian Man. With 20 Joints of Kinect [3].

Using the anthropomorphic data shown in table (1), the skeleton is divided into 9 segments: The trunk(1 segment), upper arms (2 segments), forearms (2 segments), thighs (2 segments), and shanks (2 segments). Hands and feet segments are neglected because of their relatively small fractional body masses.

The Kinect sensor provides the locations of the 20 joints in 3D space, they are used to calculate the center of masses of the 9 segments according to table (1). For example, the center of mass of the trunk is the point (X, Y, Z), where:

$$X = \text{hip center (x)} + [\ \text{shoulder center(x)} - \text{hip center(x)}\ ] * 0.54$$

$$Y = \text{hip center (y)} + [\ \text{shoulder center(y)} - \text{hip center(y)}\ ] * 0.54$$

$$Z = \text{hip center (z)} + [\ \text{shoulder center(z)} - \text{hip center(z)}\ ] * 0.54$$

The distance between the shoulder center and the hip center is multiplied by 0.54 then added to the hip center to locate the center of mass. The same calculation applies for the

www.manaraa.com

rest of the segments by only changing the fractional location from the proximal to the end point. This is a snippet of the code that calculates the center of mass of the skeleton of the user by using the locations of the Kinect sensor's joints:

```
Point COM1 = new Point(xx[0] + (xx[2] – xx[0]) * 0.54,
yy[0] + (yy[2] – yy[0]) * 0.54);
Point COM2 = new Point(xx[8] + (xx[9] – xx[8])*0.436,
yy[8] + (yy[9] – yy[8])*0.436);
Point COM3 = new Point(xx[4] + (xx[5] – xx[4])*0.436,
yy[4] + (yy[5] – yy[4])*0.436);
Point COM4 = new Point(xx[9] + (xx[10] – xx[9])*0.682,
yy[9] + (yy[10] – yy[9])*0.682);
Point COM5 = new Point(xx[5] + (xx[6] – xx[5])*0.682,
yy[5] + (yy[6] – yy[5])*0.682);
Point COM6 = new Point(xx[16] + (xx[17] – xx[16])*0.433,
yy[16] + (yy[17] – yy[16])*0.433);
Point COM7 = new Point(xx[12] + (xx[13] – xx[12])*0.433,
yy[12] + (yy[13] – yy[12])*0.433);
Point COM8 = new Point(xx[17] + (xx[18] – xx[17])*0.433,
yy[17] + (yy[18] – yy[17])*0.433);
Point COM9 = new Point(xx[13] + (xx[14] – xx[13])*0.433,
yy[13] + (yy[14] – yy[13])*0.433);
```

The code calculates the center of masses of the nine segments using the data provided in table (1). For instance, COM3 is the location of the center of mass of the left upper arm in XY coordinates, the difference between xx[4] and xx[5] (which are the locations of the left shoulder and left elbow in X respectively) is multiplied by 0.436 which is the

www.manaraa.com

correspondent data of the upper arm in table (2-1). Figure (2-11) shows an image of a skeleton taken from Kinect where the red points represent the center of masses of the 9 segments, and the white points represent the 20 points of the skeleton. The main center of mass of body is determined depending on the weighted average based on the fractional body mass of each segment, each center of mass is multiplied by its fractional body mass. Then, the results are added to each other and divided by the sum of fractions as in equation (2) where $r_{cm}^{\rightarrow}$ is the location of the center of mass in XYZ for the whole body.

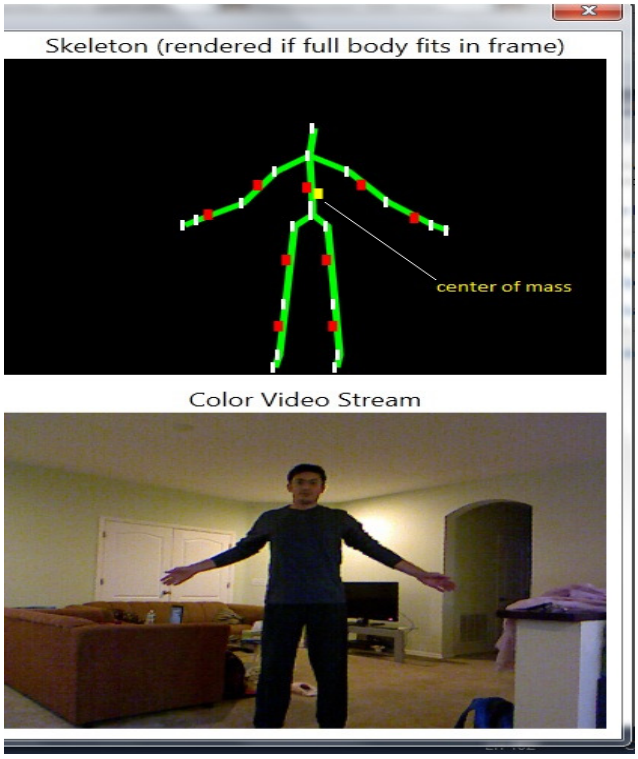


Figure 2-11: COM of Each Segment in the Skeleton.

The yellow point in figure (2-11) is the center of mass of the body according to the calculations of the 9 segments which are represented in the red color. The Kinect sensor provides continuous information about the locations of the skeleton's joints, which means that the center of mass will be continuously changing according to the user's movements.

www.manaraa.com

Chapter 3: NAO Robot

3.1 Introduction

The objectives of the work presented in this chapter are:

- Applying human motion tracking with NAO using the Kinect sensor as an interface between the user and the robot.
- Having NAO balance on one foot using the principle of the center of mass when the human similarly standing on one foot.

3.2 Background

3.2.1 NAO

NAO is a programmable humanoid robot that has a body of 25 degrees of freedom, two cameras, four microphones, two IR emitters and receivers, one inertial board, nine tactile sensors, and eight pressure sensors [11]. Figure (3-1) shows NAO's body parts, joints and sensors. NAO is equipped with tactile sensors on the head and the wrists, infrared sensors and cameras in its head. NAO is also equipped with speakers and microphones to communicate with users.

The figure also shows NAO's body joints: the head joint, the shoulder joints, the elbow joints, the wrist joints, the hip joint, the knee and ankle joints.

www.manaraa.com

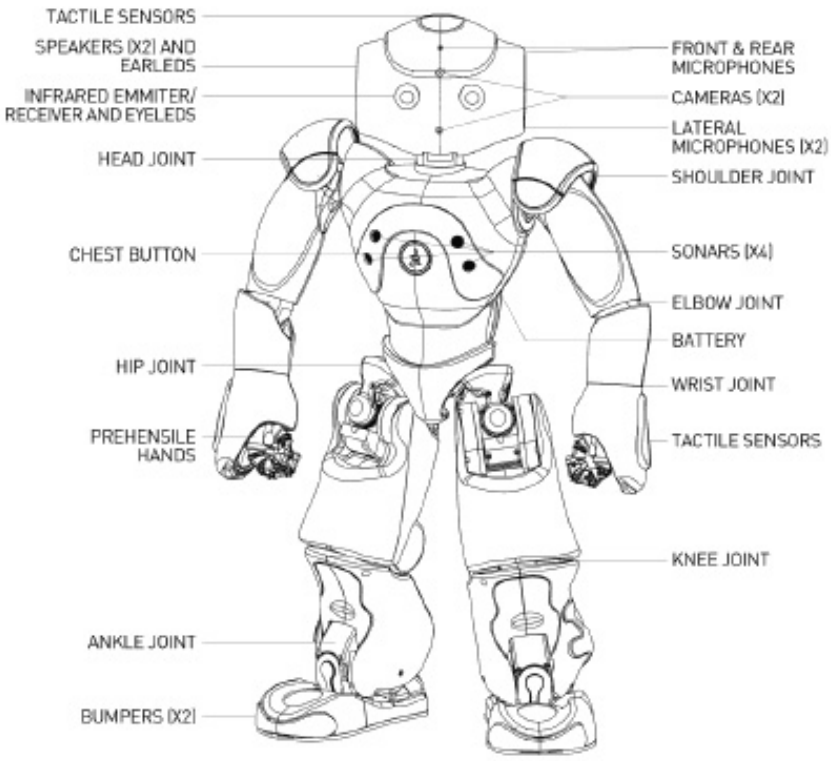


Figure 3-1: NAO V3.3. Parts, Joints, and Sensors.

NAO was developed by a French company called Aldebaran Robotics in 2004 for academic purposes. More than 1000 have been sold to universities worldwide to be used in research [12]. NAO can walk, sit, play soccer, hear, talk, feel and perform different poses with its 57 cm tall body, and because of these abilities, it is being used widely in medical fields mainly with sick children to cheer them up and specially with autism treatment in children [12].

NAO was developed to perform some remarkable routines like performing the "Tai Chi" dance where it shows the robot's ability to keep its balance on one leg, or narrating a short story for entertaining purposes.

www.manaraa.com

3.2.2 NAO V3.3 Joints

The following figures and table show how much freedom NAO's arms and legs have in degrees and radians. Table (3-1) and figures (3-2, 3-3, and 3-4) are taken from NAO Software 1.12.3 documentation [13].

Figure (3-2) shows the left and the right arm of the robot. The robot can roll its shoulders in a full 94 degrees and elbows in 88.5 degrees.

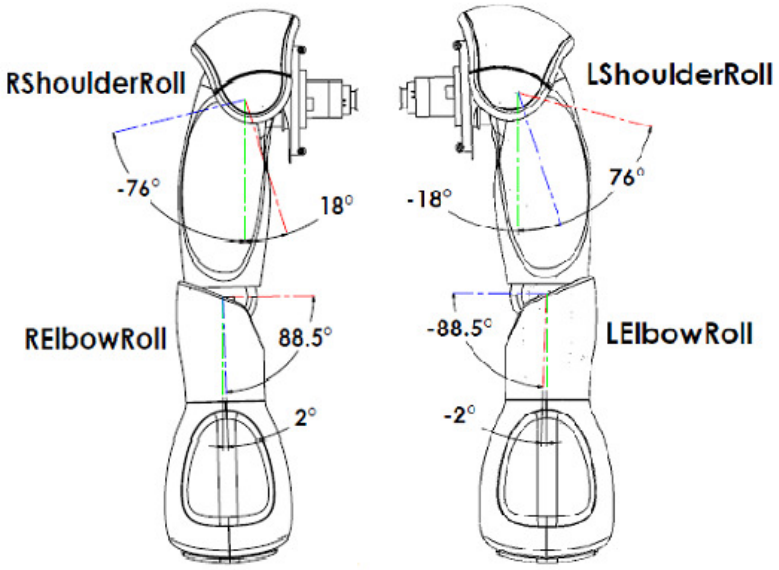


Figure 3-2: NAO's Arms Movement (Shoulder Roll).
From NAO Software 1.12.4 Documentation [13].

There are some limitations in NAO's arms movements according to the figure above, especially in its elbows, but the left and right shoulders roll fairly well compared to human shoulders.

Figure (3-3) shows another movement of the NAO's arms. The left and right arms can rotate 239 degrees around their axis.

www.manaraa.com

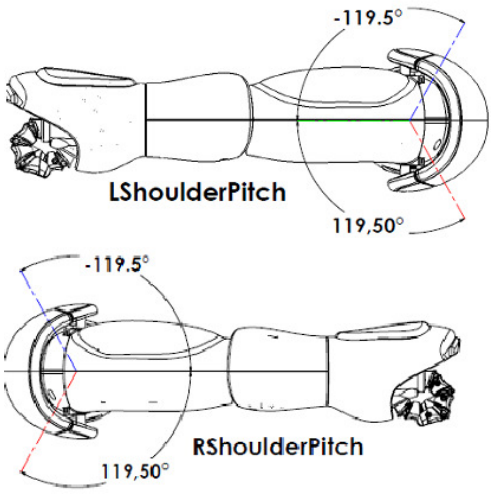


Figure 3-3: NAO's Arms Movements (Shoulder Pitch).
From NAO Software 1.12.3 Documentation [13].

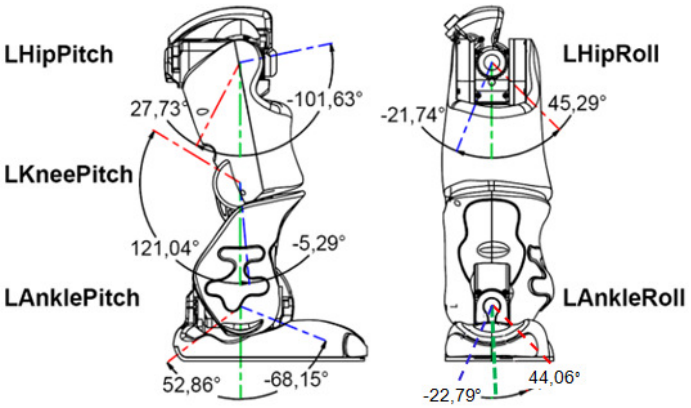


Figure 3-4: NAO's Left Leg [13].

NAO's legs move in a good range as shown in figure (3-4) which allow it to perform different maneuvers and positions. These ranges give NAO the ability to walk, sit, and play soccer in a fairly human-like way. Note that NAO's right leg has the same ranges.

www.manaraa.com

Table (3-1) shows the ranges of both arms' motors in degrees and radians. NAO's elbows can go from full stretch to 88.5 degrees, however there is some limitation in elbows bending comparing to the human elbow which can go more than 90 degrees.

Table 3-1: The Range of Left and Right Arms' Motors [3].

Joint Name	Motion	Range (degrees)	Range (radians)
LShoulderPitch	Left shoulder joint front and back (Y)	-119.5 to 119.5	-2.0857 to 2.0857
LshoulderRoll	Left shoulder joint right and left (Z)	-18 to 76	-0.3142 to 1.3265
LElbowRoll	Left elbow joint (Z)	-88.5 to -2	1.5446 to 0.0349
RshoulderPitch	Right shoulder joint front and back (Y)	-119.5 to 119.5	-2.0857 to 2.0857
RshoulderRoll	Right shoulder joint right and left (Z)	-76 to 18	-1.3265 to 0.3142
RElbowRoll	Right elbow joint (Z)	2 to 88.5	0.0349 to 1.5446

3.2.3 NAO's Sensors

NAO is equipped with different types of sensors which allow the robot to explore the environment around it, and avoid obstacles:

- Contact sensors, which are distributed on its feet, head, and hands. The sensors on its feet are located at the tip of each foot (bumpers), which alarms NAO that it has hit an object or an obstacle with its feet. The other sensors are tactile sensors, which are located in NAO's head (three sensors) and the hands (three sensors), and they allow the robot to sense objects with its hands.
- Force sensitive resistors are located at the bottom of NAO's feet, each foot has four resistors. The values of the resistors change according to how much

www.manaraa.com

pressure is applied on NAO's feet, these values help the robot to distinguish whether it is standing on its feet or not, and to redistribute its body weight while it is walking. The sensors' working range is 0 N~25 N [13].

- The sonar, which is two ultrasonic sensors (two transmitters, and two receivers), that are located on NAO's chest, they are used to detect obstacles and objects in a 0.25 m ~ 2.55 m range with a resolution of 1 cm [13].

3.2.4 Coding

The coding language that is used in the control NAO is C sharp. It is a very efficient high level language that was developed by Microsoft. The Kinect sensor works perfectly with this language since Kinect is also a Microsoft product. NAO is also supported with this language that runs on dot NET framework. The code of the Kinect sensor starts with some definitions for initializing the sensor's parameters, like the RGB, depth, and skeleton streams. The depth stream and the skeleton stream work together to locate the body's joints in 3D space. The skeleton consists of 20 joints, each joint represents a specific location on the human body: the head, the shoulder center, left and right shoulders, left and right elbows, left and right wrists and hands, the spine, hips, left and right knees, left and right ankles and feet.

The skeleton is built when the user's body is tracked, each point of the skeleton is mapped to a depth image to represent the body in 3D space, the X and Y data are expressed in pixels, and Z data are expressed in centimeters. Each point also has a fourth value, W, which represents the point's confidence level of a value from 0 to 1, the 0 value indicates that the joint is not clearly visible for the sensor and its position may not be correct, the 1

www.manaraa.com

value indicates that the joint's position is visible. The differences between the joints in X and Y directions represent the distances between the joints in centimeters.

3.3 Mimicking Human Arm Motions in a Humanoid Robot

The previous section of this chapter introduced the humanoid robot NAO and showed its features and abilities. This section discusses the developed control strategies used so that the NAO could track a human arm's motions and to balance on one leg using the Kinect sensor as an interface.

As discribed in chapter two, the Kinect sensor has the ability to build a skeleton from the depth image of a human body. This skeleton with detected joints will be used to calculate angles between the upper arms and the torso, and between the upper arms and the forearms of a human. The remaining angles can be calculated from the positions of leg joints, like the knee angle.

NAO's body parts can be controlled by setting angles for both arms and legs that were shown in figures 3-2, 3-3, and 3-4. The idea is to use measured angles of human body parts to control NAO's movements, in other words, NAO will mimic the user's movements. NAO, on the other hand, is supported by coding libraries that enable users to access the robot's actuators. The Kinect sensor will work as an interface between the user and NAO who can be in two different places.

Creating the skeleton and having all data of each point makes the calculation of angles straightforward. The next part of the code calculates the angles needed to control NAO with the Kinect sensor. 3D vectors are created from shoulders, elbows, and hands positions in order of calculate the angles [see appendix A].

www.manaraa.com

Angles between the upper arms and the X axis represent shoulder roll movements in NAO, but the angles between the upper arms and the Z axis represent shoulder pitch movements in NAO. For example, Figure (3-5) shows two 3D vectors: the left shoulder vector (1) and the left elbow vector (2), the angle between it and the X axis (1, 0, 0) and vector number 2 will be the complementary angle of the (left shoulder roll) angle in NAO. Safety conditions are added to the code to protect the robot, like making the maximum value of the left shoulder roll 76 degrees instead of 90 degrees, because NAO cannot roll its arms in the XY plane more than 76 degrees while a human can. For comparison, a protractor was used to measure a human's upper extremities' ranges and the results were: the shoulder roll range is about 120~150 degrees, and the shoulder pitch is 250~260 degrees. NAO's full shoulder roll range is 94 degrees, and shoulder pitch is 239. Note that the human has a larger range than the NAO in these and most other joints.

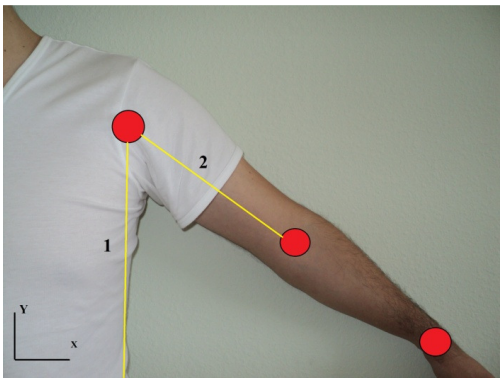


Figure 3-5: Calculating the Angle of the Left Shoulder.

The following flowchart shows how the code receives the information from the Kinect sensor, processes it and sends it to NAO. The yellow blocks indicates to the Kinect SDK and the blue blocks indicates to the NAO SDK.

www.manaraa.com

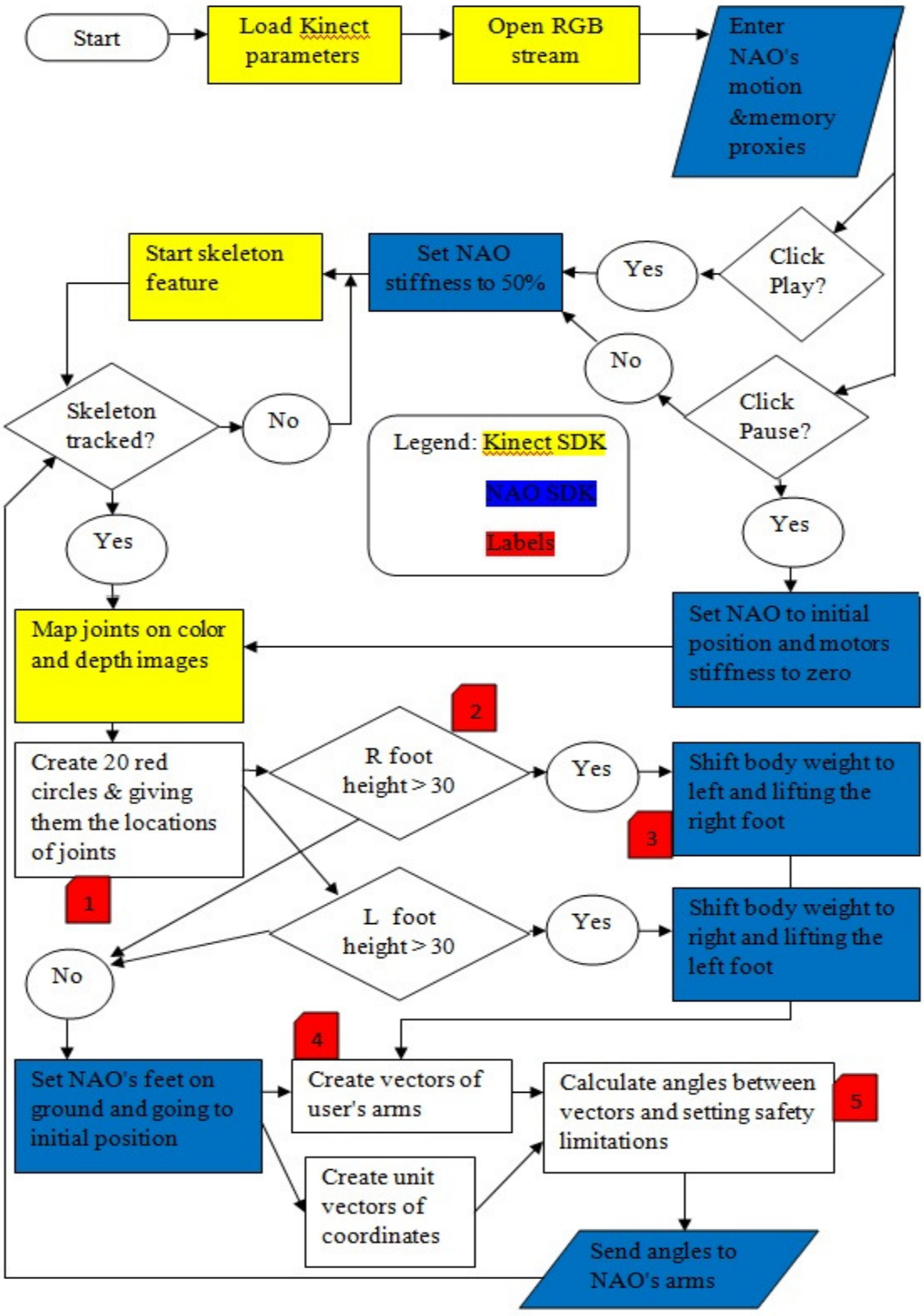


Figure 3-6: The Flowchart of NAO Code

www.manaraa.com

The code starts with setting Kinect's parameters like the RGB, depth and skeleton features. The yellow blocks represent Kinect's SDK and the blue blocks represent NAO's SDK. The Kinect sensor's smoothing parameters are set to 30% to reduce the jittering of the joints' positions. NAO motion and memory proxies are provided after setting Kinect's parameters. NAO's motors stiffness will be set to 50% of the full power (which is enough to move the NAO's extremities) to reduce jerkiness in NAO's motions. When the skeleton is tracked, the joints provided by Kinect will be mapped on the color and depth images, the next block (labeled with 1) creates 20 red circles on the joints' locations to make them visible to the user. Block number 2 tests the height of the user's foot above the ground, if the height is more than 30 cm, NAO will do the routine in block number 3 (which will be explained later in section 3.4). Block number 4 shows the process where the vectors of the users arm joints are created. Block number 5 shows the calculation of the angle which depends on the dot product between the two vectors is calculated by:

$$a \,.\, b = |a||b| \cos\theta \qquad \text{Equation (7)}$$

where θ is the angle between the two vectors. Then, the angle is converted to radians to use it directly with NAO's code.

To explain the procedure of calculating the arm angles, here is a snippet of the code showing the calculation of the left upper arm roll angle:

```
kinectshoulderleft = new System.Windows.Media.Media3D.
Vector3D(xx[4], h[4], (zz[4] *   100));
kinectelbowright = new     System.Windows.Media.Media3D.
Vector3D(xx[9], h[9], (zz[9] *   100));
shoulder_left1 = kinectshoulderleft – kinectelbowleft;
```


www.manaraa.com

```
System.Windows.Media.Media3D.Vector3D newx = new
System.Windows.Media.Media3D.Vector3D(1, 0, 0);
angleshoulderleft1 =  System.Windows.Media.Media3D.
Vector3D.AngleBetween(newx, shoulder_left1);
motion.setAngles("LShoulderRoll",(float)(angleshoulderleft1
), 0.1f);
```

The first two lines show the starting and end point of the left upper arm vector where the left shoulder and left elbow joints are used, the depth location of these joints is in meters and multiplied by 100 to be converted to centimeters. Vector "`newx`" is the unit vector of the X coordinate. The variable "`angleshoulderleft1`" represents the angle between the left upper arm vector and "`newx`" vector, the code provides a direct way to find the angle between vectors in the command "`AngleBetween`". The angle is in degrees and must be converted to radians before sending it to NAO. After that, the code sends the angle to NAO through the motion proxy with the command "`setAngles`".

3.4 Mimicking a Human Standing on One Leg

The second objective is testing NAO's ability to balance on one foot using its center of mass. The Kinect sensor will be used to detect any significant vertical change in the user's feet positions, if the user lifts one foot more than 30 cm off the ground, NAO will lift the same foot and balance on the other. Because of the differences in weight distribution between NAO and human, the NAO cannot perfectly mimic the motions of a human while balancing on one foot. Figure (3-7) shows NAO's three phases of lifting the right foot, in (A) NAO is standing in its initial position which is the first phase, the red arrow indicates that NAO's center of mass is the supported area between its feet, the

www.manaraa.com

second phase is shifting the center of mass to the left leg as represented in (B) from the same figure, the third phase is lifting the right foot.

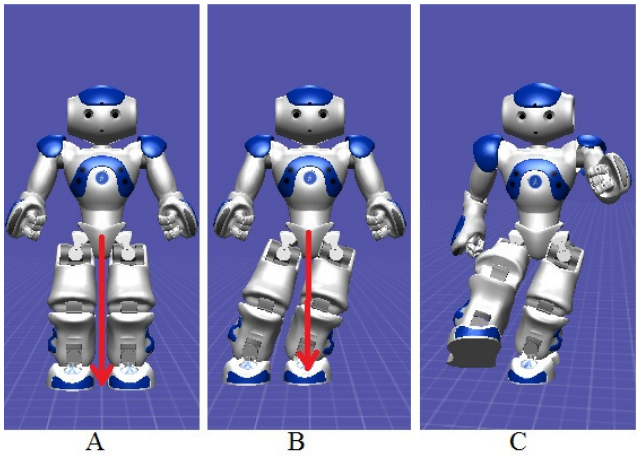


Figure 3-7: NAO is Lifting the Right Foot.

In the initial position, the distance between the positions of NAO's feet and its torso in the X axis is 5 centimeters. This means that the torso should be shifted 5 cm to the left or right in order to be over the left or right foot respectively. Setting the angles of left-hip-roll and right-hip-roll shown in figure (3-4) and for +19 degrees or -19 degrees makes the position of the torso above one of the feet. Notice that when NAO is at the initial position, the X location of its center of mass is the same X location of the torso, it is important that NAO starts from the initial position.

The code for lifting NAO's foot is described as follows:

- The Kinect sensors detects if the user's foot is higher than 30 cm above the ground.
- Setting the angles of both hips to 0.33 rad and both ankles to -0.33 rad (shifting to the right), or the angles of the hips to -0.33 rad and ankles to 0.33 rad (shifting to the left).

www.manaraa.com

- Pause for 1.5 second to regain the balance at this pose, and eliminate jerks.
- Lifting the required foot about 70 degrees, and bending the knee to the back, to prevent the foot from hitting the ground when it goes back to the initial position.

Note that NAO does not mimic the exact motion of the user's leg; it only reacts with the motion by lifting the correspondent leg because the mass distribution on NAO is not the same as a human.

NAO can go back to the initial position by reversing these steps, the lifted foot goes back from 70 degrees to the initial location (NAO's leg angle at the initial position is 25 degrees), then pausing for 1.5 second to balance back on two feet, then shifting the hips and angles back to their initial angles (0 degree). The speed of these phases can be controlled by setting the fraction of NAO's motors maximum speed.

3.5 Results

Kinect sensor provides an excellent skeleton presentation for the human body which allows developers to use it in many research fields, the sensor is considered a huge step in controlling robots wirelessly. On the other hand, NAO has the ability to move and act like a human and it can be controlled easily by accessing its features. The code that was explained previously combines the quality of the depth image of the Kinect and the definition of NAO's actuators. The Kinect sensor gives the skeleton data, which are then used to determine the angles of the left and right arms, after that, they are implemented in NAO's arms movements. This method saves the trouble of trying to downsize the length of a human arm into NAO's relatively short arms. Figure (3-8) includes eight situations of NAO with the user.

www.manaraa.com

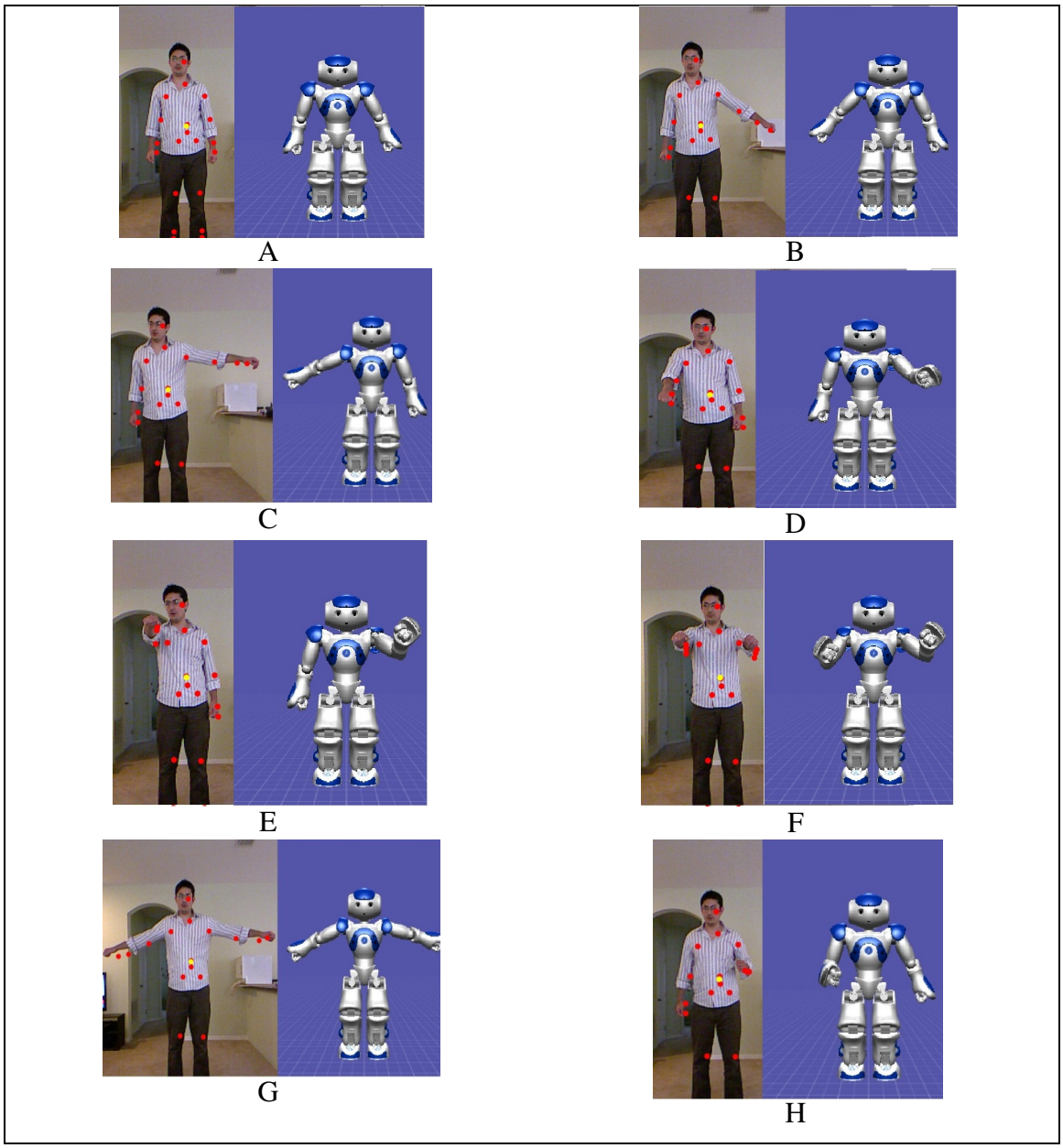


Figure 3-8: NAO Mimicking the User's Motion.

Figure (3-8 A) represents NAO in its initial position, pictures B and C show the NAO's reaction to the user's left arm, where the angle of the left arm is calculated and used with NAO correspondent arm. Picture D and E show the pitch angles of the right and left arms

www.manaraa.com

respectively. Pictures F and G represent the reaction of NAO with two arms together. And picture H shows NAO bending its left elbow with the user.

Figure (3-9) shows NAO lifting its left foot by shifting its body weight to the right.

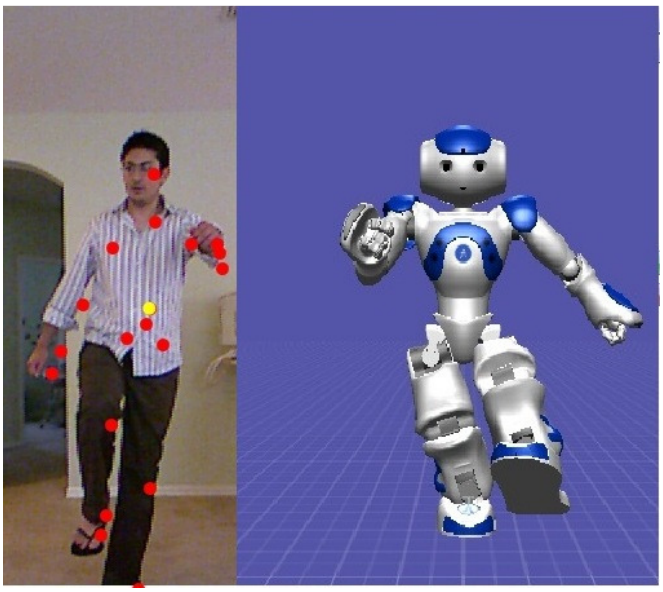


Figure 3-9: NAO Lifting its Left Foot.

We can also see that the user's right leg is supporting his center of mass (represented in yellow) which enables him to balance on one foot. Figure (3-9) also shows how the robot is still able to mimic the arm movements while standing on one leg.

3.6 Conclusions

Applying the user's arm motion tracking was successfully done on NAO with some limitations caused by NAO's arm structure and the safety factors in the code [see Appendix A] and the Kinect sensor was successfully used as an interface between the user and NAO. There was a time delay in following the arm and leg movements, and that time varies depending on how fast the skeleton is being detected. For the second objective, NAO was able to balance on one foot and follow the human legs by lifting its

www.manaraa.com

foot when the user lifts his foot more than 30 cm above the ground, however, the robot does not mimic the user's leg movement since the structure of NAO's leg is not like the human's. The principle of the center of mass was used to balance NAO on one foot by repositioning its center of mass over the supporting leg.

www.manaraa.com

Chapter 4: Robot Assisted Balance Training

4.1 Introduction

This chapter describes how the Kinect can be used as an assessment and feedback tool to an individual using the robot-assisted balance training. The objectives of this chapter are:

- To calculate the center of masses of patients undergoing the robot-assisted balance training using the Kinect sensor to study their weight distribution during the training.
- To improve the visual feedback of the balance training by showing patients useful information about their leaning techniques.
- To increase the efficiency of the training program by making patients learn how to shift their body weights properly.

4.2 Background

4.2.1 Rehabilitation

The rehabilitation process is a medical activity that helps people who have experienced a serious injury, illness, or surgery to recover their physical and/or mental skills and capabilities. The rehabilitation can be physical to help the patients regain their mobility, strength, or balance. It can be a speech-language pathology, for the people who suffer from speaking, reading, and/or writing disabilities. Sometimes, the rehabilitation process can be psychiatric to help patients recover their mental capabilities [16].

www.manaraa.com

People who have experienced serious strokes are highly susceptible to physical disabilities depending on the location of the damage in the brain and how severe the damage is. For instance, if the damage happens in the part that controls the balance and coordination in the brain, the patient may suffer from impaired balance in standing or walking.

Most patients following stroke undergo rehabilitation programs to recover. The recovery processes can be very slow in most of the programs, and not so efficient if patients cannot adapt with them. In order to have a fast and more efficient rehabilitation program, a new process was introduced called rehabilitation robotics which aims to optimize the physical rehabilitation by using robotic technologies in the process. Rehabilitation robots have many advantages like efficiency and the recovery time especially in treating balance impairment.

The robot-assisted balance training (RABT) is a training program that improves the recovery of stroke patients who suffer from impaired balance in standing or walking. The program returns a visual feedback to patients in order to help them improve their walking patterns. The training program applies an external pulling force on the healthy sides of patients to force them to shift their body weights towards the affected sides. The visual feedback shows them how much weight they should put on one foot in order to complete a proper move. The purposes of this training program are: to help patients to develop their balance control during the training and to learn how to shift their center of gravities to different postures without losing their balance, which is the precursor to being able to effectively walk.

www.manaraa.com

4.2.2 Device Components and Procedure

Figure (4-1) shows the physical components of the training program. The training program was developed by Dr. Seok Hun Kim, an Assistant Professor in the School of Physical Therapy and Rehabilitation Sciences in the University of South Florida, and was built in the Rehabilitation Engineering and Electromechanical Design Laboratory by Dr. Kyle B. Reed, an Assistant Professor in the Department of Mechanical Engineering in the University of South Florida.

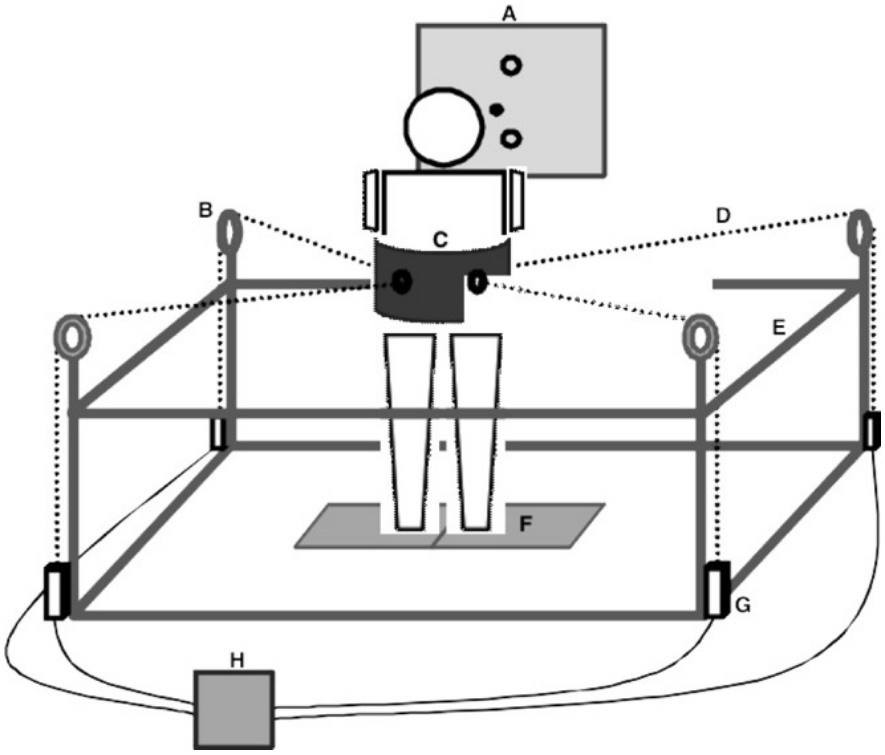


Figure 4-1: Robot Assisted Balance Training[14].
A. monitor, B. pulley, C. waist built, D. pulling cable,
E. frame, F. Wii balance board, G. motor, H. control box.

The patient will be wearing a waist belt and a safety harness during the training session; the waist belt is attached to two motors through two pulleys with two ropes, two force

www.manaraa.com

measuring devices are mounted on the ropes. The two motors and the force sensors are controlled by a computer. The computer reads the values of the force sensors and sends signals to the two motors to pull the ropes to maintain a constant tension [14].

The two motors apply pulling forces up to 60 Newton to the patient's pelvis while standing on the balance board. The patient is asked to shift his/her body weight and pay attention to the monitor to watch how much pressure is applied on the affected side. After reaching the specified amount of weight on one foot, the patient has to take a step forward, then step backward with the same procedure. The trials are repeated up to 100 times. The patient is expected to show a significant adaption in his/her walking pattern after the experiment, this forces the patient to use his/her affected side more often for a brief period. The patient might show more similarity in his/her walking and balance control pattern.

The robot-assisted balance training depends on how the patient moves his/her center of pressure towards the affected side. Shifting the center of pressure to one side properly requires the patient to also shift his center of mass over the affected leg in order to be prepared to take a step forward or backward. Preliminary results with healthy individuals showed little adaptation when subjects leaned incorrectly, but showed beneficial after effects when leaning correctly. The after effects manifest as an asymmetric step pattern when walking over ground, which was tested before and after training on the RABT.

www.manaraa.com

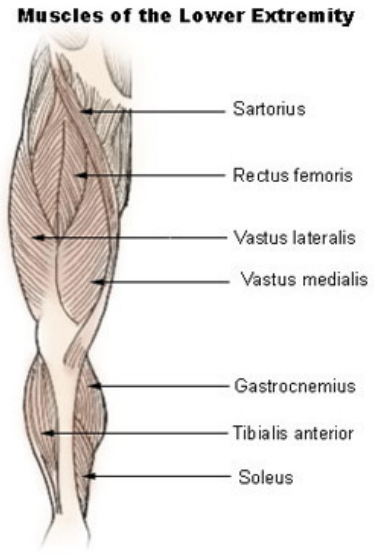


Figure 4-2: Vastus Lateralis [15].

Patients should counter the external pulling force applied on them by leaning properly to the other side to train their "vastus lateralis" muscles shown in figure (4-2). But, incorrect leaning could limit muscles improvement during the training, train the wrong muscle, or sometimes develop wrong walking and balance control patterns.

4.3 Using the Kinect as Feedback about Proper Leaning Technique

The previous section introduced the robot-assisted balance training as a physical rehabilitation program for post-stroke individuals that helps them regain their balance. The objectives of this work are:

- To calculate the center of masses of individuals using the Kinect sensor to study their weight distribution during the training.
- To provide visual feedback of the balance training by showing patients useful information about their leaning techniques.

www.manaraa.com

- To increase the efficiency of the training program by teaching patients how to shift their body weights properly.

Microsoft Kinect sensor can be integrated as an enhancing technique in the training program by returning another feedback for the patient or the supervisor indicating that the patient is leaning properly or not. As mentioned before, Kinect has the ability to detect human's joints and construct a skeleton based on them. Chapter two explains how the center of mass of a human body can be calculated using the Kinect sensor. This center of mass can be used to determine whether the patient (undergoing the RABT) is doing the training correctly or not.

The patient is required to shift his/her center of mass above the position of the affected leg. In order to do that, the patient must shift his/her hip only and keep the trunk upright to counter the external pulling force. The balance board that is used in the training program returns raw data of the center of pressure of the patient, but the center of pressure may not be at the same location of the center of mass of the body because the center of pressure is influenced by the amount of force the patient applies on each leg while shifting to either side. Also, the center of pressure does not significantly change between a correct lean and an incorrect lean.

4.3.1 The Need for a Measure of Leaning

The initial balance training only took measures of the individual's center of pressure using two balance boards. However, as I will show here, the data is insufficient to determine if the individual is leaning correctly. Figures (4-3 and 4-4) show the data of the center of pressure taken from the balance board: Figure (4-3) shows the change of the center of pressure while leaning to the left correctly. The x axis represents the operating

www.manaraa.com

time of the balance board in seconds, the y axis represents the shifting distance the center of pressure. The raw data of the balance board are converted to the real world coordinates (seconds and centimeter) as follows: The distance between the user's feet while standing on the board is 30 cm, while the raw data of the board range from 1 to -1, each point of the data is multiplied by 15 to get a range from 15 cm to -15 cm. The raw data is taken at 60 points per second speed, so the time base is divided by 60 to distribute the 900 points on the operating time (which is 15 seconds).

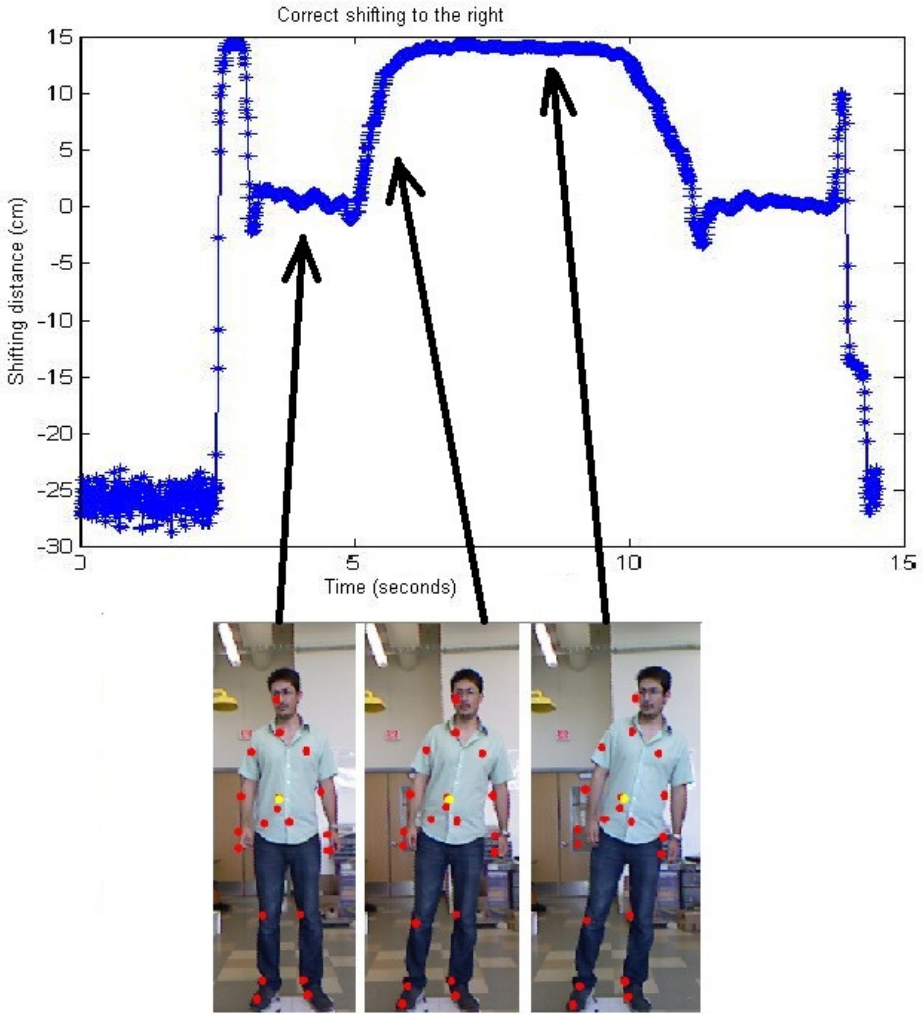


Figure 4-3: The Center of Pressure of Leaning to the Left Properly.

www.manaraa.com

The first two seconds in figure (4-3) represent a random set of data (around -25 cm) since the user is not standing on the balance board. At (2.5 sec) the user steps on the balance board with his right foot and we can see the output data is changing significantly up to 15 cm, where the user's full body weight is concentrated on the right foot at that moment. Then the readings goes back to the center point (0 cm) when the user puts the other foot on the board. The pressure of the body starts shifting from the center to the right until it reaches 15 cm at the point where the body reaches the furthest position to the right, then goes back to the center again. At the 14 second point, the user steps down from the balance board with his left foot.

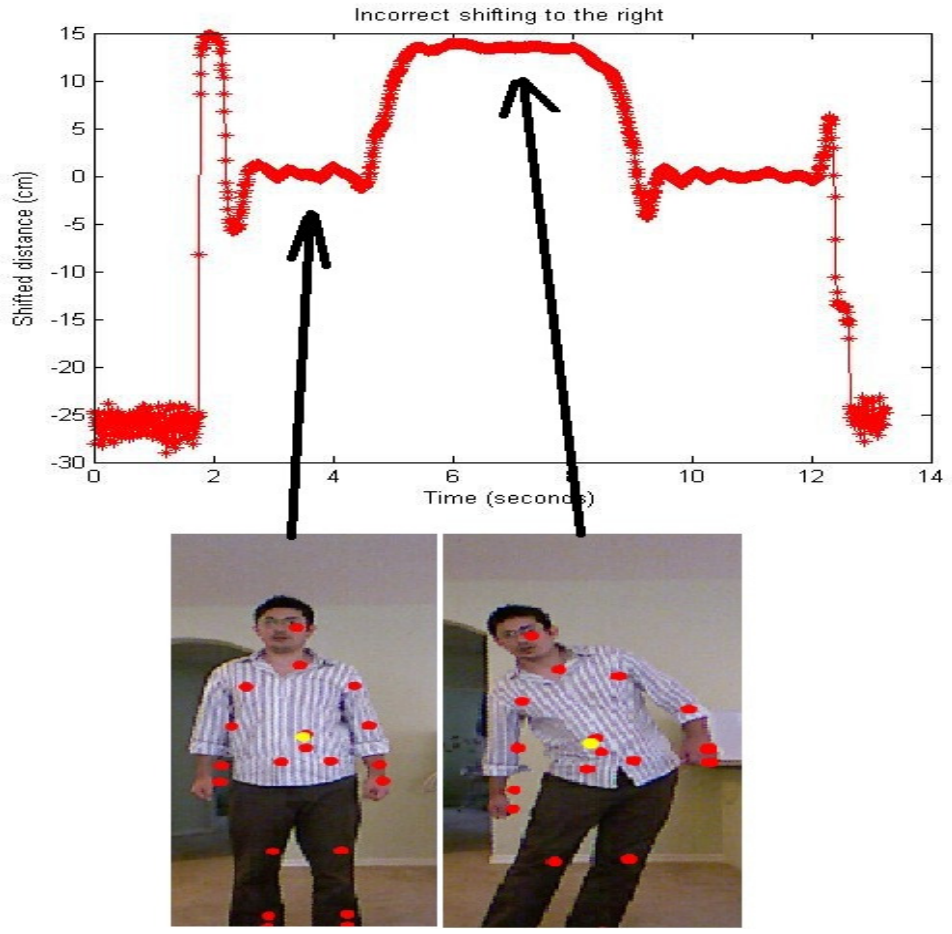


Figure 4-4: The Center of Pressure of Leaning to the Right Improperly.

www.manaraa.com

Figure (4-4) also shows the user's center of pressure while leaning to the right side incorrectly, the pressure builds up until it reaches the maximum, then returns to the center again. Both figures are very similar in the way the center of pressure shifts to the maximum location which is 15 cm in both cases. The figures show that the center of pressure was shifted to the extreme right but they do not give a clear idea about whether the leaning was correct or not.

Because of the similarity between these two figures, the feedback monitor will always show the patient that he/she has reached the desired amount of pressure regardless of the way the patient leaned. That will cause a mix between the correct and incorrect movements, which means having a wrong training technique, which is not effective for training.

4.3.2 Providing Feedback about Leaning

This problem could be solved by assigning a specialized person who teaches the patient how to lean and take a step properly. After that, the patient may learn how to perform the rest of the training correctly, or may develop an incorrect training after he/she feels tired during the training. A better way to solve that problem is to use the Kinect sensor with the system to give a feedback to the patient the entire training session. The sensor will be applied in front of the training area facing the patient and will be connected to the feedback monitor. The patient will notice two horizontal bars (one for the left leaning and one for the right leaning). The following flowchart shows how the the Kinect's data are processed to the training program:

www.manaraa.com

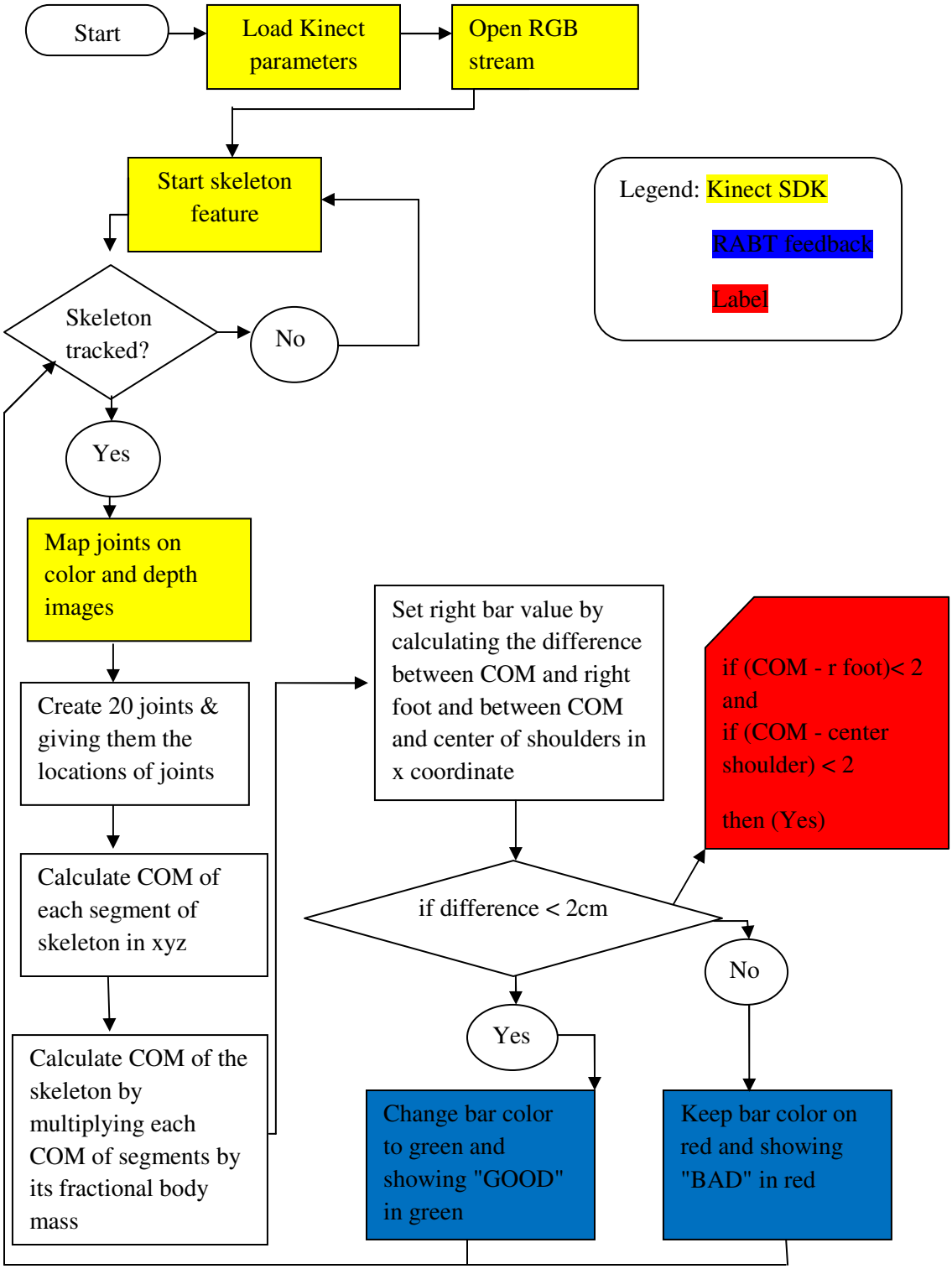


Figure 4-5: The Flowchart of RABT Code

www.manaraa.com

The red card in the flowchart explains the conditions of the bar colors. When the difference between the X locations of the center of mass and the right foot and between the center of mass and the shoulder center is less than 2 cm, the bar will be fully loaded and turn to green. When the patient starts leaning correctly to the right side for instance, the Kinect sensor calculates the difference between three points: his/her center of mass (which is explained in chapter 2), the position of the right foot, and the position of the midpoint between the left and right shoulders locations (the shoulder center). When the difference between the three points is less than 2 cm, it means that the three points are on the same vertical imaginary line. As the patient approaches the correct position, the right side bar starts loading in red color, and when the patient reaches the wanted position correctly, the bar will be fully loaded and will change into green, a green "GOOD" will illuminate on the monitor in front of the patient.

Here is a snippet of the code that shows the calculations of the bars' values:

```
progressBar1.Value = 30 - (COM - xx[15]);
progressBar2.Value = 30 - (xx[19] - COM);
if ((30 - (COM - xx[15])) > 28 && (30 - (xx[2] - xx[15])) > 28)
   {
   progressBar1.Foreground = new SolidColorBrush(Colors.Green);
   label1.Foreground = new SolidColorBrush(Colors.White);
   label2.Foreground = new SolidColorBrush(Colors.Green);
   }
else
   {
    progressBar1.Foreground = new SolidColorBrush(Colors.Red);
```


www.manaraa.com

```
label1.Foreground = new SolidColorBrush(Colors.Red);
label2.Foreground = new SolidColorBrush(Colors.White);
}
```

The value 30 represents the value of the bar, when the difference between `COM` and `xx[15]` (the left foot) reaches its maximum value (15 cm), the bar's value will be 30 - 15 where the bar is half loaded. Notice that " `label1`" represents the word "BAD" and "`label2`" represents the word "GOOD". When the difference is less than 2 cm, the bar (which is fully loaded) will turn to green, the word "GOOD" illuminates in green and "BAD" disappears. If the difference in more than 2 cm, the bar's color remains red, the "GREEN" label disappears and the "BAD" label illuminates in red.

The following figures represent different postures of the user in front of the Kinect sensor. Figure (4-6) shows the positions of the joints while standing upright, the red points represent the 20 joints of the skeleton, and the yellow point represents the center of mass of the body which can change according to the joints' positions. The center of mass in this figure is located on the trunk right above the hip point.

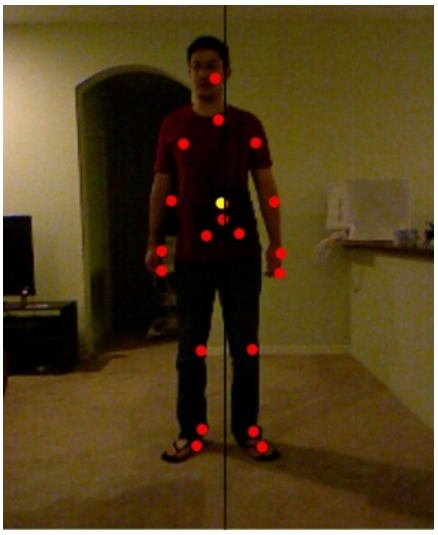


Figure 4-6: Standing Upright.

www.manaraa.com

Figure (4-7) is an example of leaning to the right correctly, the center of mass (the yellow point) is located above the right foot point, the trunk is still upright, and the shoulder center point is also aligned with the center of mass and the right foot on the white line.

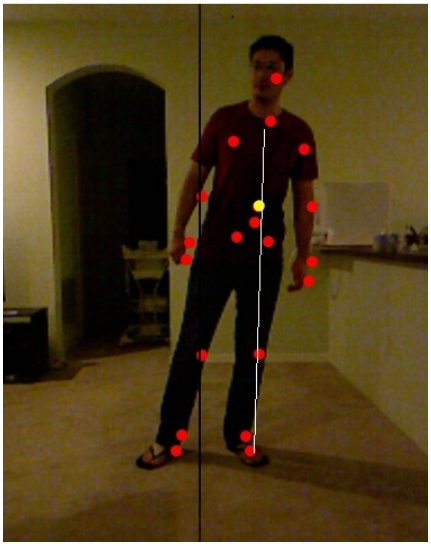


Figure 4-7: Leaning to the Right Correctly.

At this position the patient will notice the right foot bar changing into green and a green "GOOD" illuminates. The patient can now take a step forward or backward and continue the training.

Figure (4-8) shows different positions of inappropriate leaning. In (4-8 A), the hips are shifting to the right and the center of mass is located roughly above the right foot, but the upper body part is bending to the opposite direction; it is the most common mistake being done by patients because they feel more balanced at this position while there is an external force bulling them the other way. (4-8 B) is also a bad way of leaning where the patient uses his/her upper body part to shift to one side, this position trains the wrong leg muscles, and patients may feel unbalanced doing it. The white line shows clearly that the

www.manaraa.com

center of mass might go outside the balance area between the two feet which may cause the patient to fall.

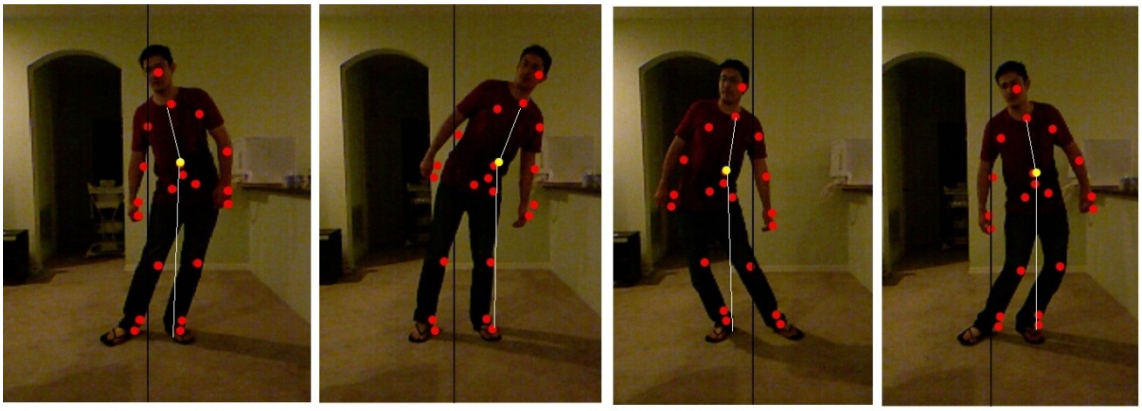


Figure 4-8: Different Inappropriate Positions.
A) Shifting the hips to the right with bending the upper part of the body to the left.
B) Bending trunk to the right without shifting the hips.
C) Shifting to the left and bending the trunk. D) Bending the knees with the leaning.

In 3-D, the hips are slightly shifted to the side, but the center of mass is still inside the balance area and the knees are bent. The patient must not bend his/her knees during the training because it might change the calibrations of the external pulling force around the waist area.

Patients can develop many inappropriate moves during the training session because they may feel uncomfortable from the pulling force or the safety harness they are wearing. But integrating the Kinect sensor with the system can improve the training efficiency and may also decrease the number of required sessions.

www.manaraa.com

The following figures show snapshots of the program. Figure (4-9) shows the correct leaning of the user, the indicator bar is fully loaded and a word "GOOD" is showing in green.

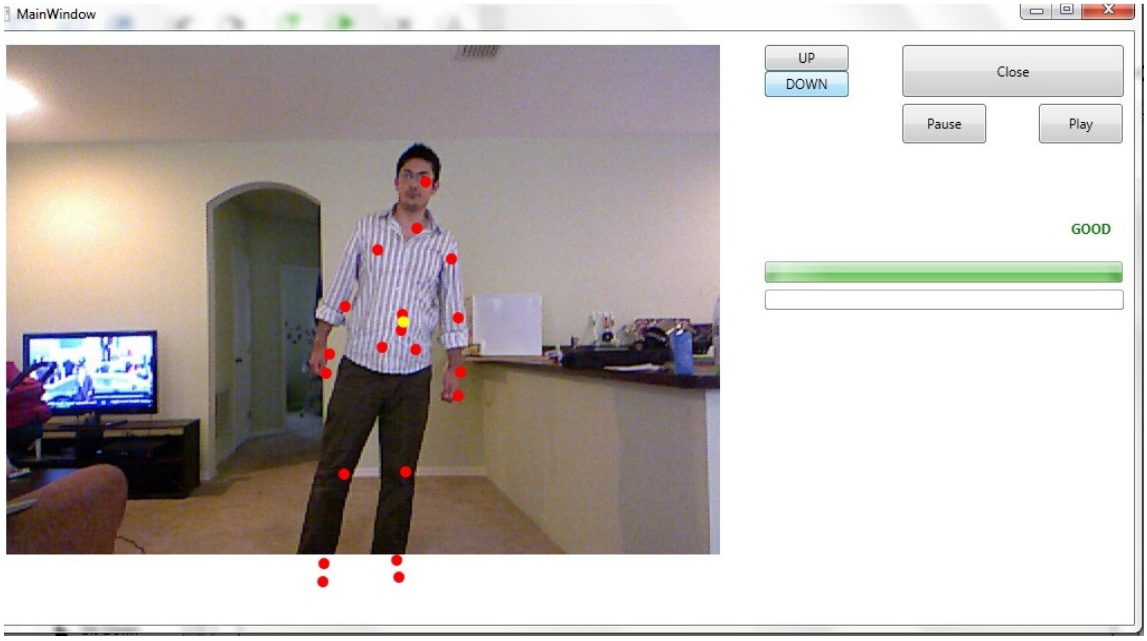


Figure 4-9: A Snapshot of the Correct Lean.

Figure 4-10: A Snapshot of the Incorrect Lean.

www.manaraa.com

If the patient reaches the wanted position and the three points are not aligned, the bar may be fully loaded but stays red and a red "BAD" illuminates on the monitor. Figure (4-10) and (4-11) show an inappropriate leaning to the left in two different ways, the indicator bar is red and a word "BAD" is showing. The user will know that his move was not correct.

Figure 4-11: A Snapshot of Another Incorrect Lean.

4.4 Results and Conclusions

This chapter showed that the robot-assisted balance training is an interesting rehabilitation process where patients can learn how to adapt their walking pattern and regain their balance. The center of mass of the patient was calculated using the Kinect sensor and was used to study the user's weight distribution during the training. The visual feedback was improved, and showed a significant improvement of the leaning techniques. The robot-assisted balance training did not previously give a clear feedback to patients who perform the training improperly.

www.manaraa.com

The Kinect sensor increased the efficiency of the balance training and showed indications of offering useful information to healthy individuals where the they can see how their center of masses are being repositioned and learn how to lean properly during the training. Also, The feedback of the Kinect sensor is affected with the distance between the sensor and the patient, and the surrounding environment of the test. Note that the Kinect sensor feedback was not tested on real patients, and the use of the Kinect sensor did not affect the training procedure, it is an addition to RABT's visual feedback.

www.manaraa.com

Chapter 5: Conclusions and Future Work

As was discussed in the introduction chapter, this thesis studied the property of the center of mass of the human body, and using it with two applications by using the Kinect sensor. The first application studied the ability of the humanoid robot NAO to track a user's arm motion and balance on one foot using the Kinect sensor. The results showed that NAO can balance on one foot by repositioning its center of mass. The second application discussed the robot-assisted balance training and the impact of the incorrect leaning on the training results. It also discussed the use of the Kinect sensor in the training program to calculate the center of mass of patients. The feedback of the Kinect sensor during the training showed a potential improvement in patients' leaning techniques.

There are several future works that may be done in order to improve the results of this thesis. Firstly, it is highly recommended to use the latest version of Kinect SDK in the future. The newest version was released in May 21, 2012, and it offers seated skeletal and facial tracking with advanced speech recognition options. Also, using two or more Kinect sensors to interface with NAO to optimize the operation of determining the arm gestures. Until this date, Kinect's software supports the skeleton feature for only one sensor, it does however support the depth feature of multiple Kinects at the same time.

www.manaraa.com

As discussed in chapter four, it will be more efficient to test the effect of Kinect on the robot-assisted balance training with real patients in the future, and compare between the results before and after using the sensor. Furthermore, improving Kinect's graphical interface that is being used with RABT, which may increase the efficiency of the feedback.

www.manaraa.com

References

1. A 3D Multi-Aperture Image Sensor Architecture Keith Fife, Abbas El Gamal and H.-S. Philip Wong Department of Electrical Engineering, Stanford University, Stanford, CA 94305-4055

2. Fairhead Harry. *All About Kinect*. 2012. Retrieved April, 2012. Retrieved from: http://www.i-programmer.info/babbages-bag/2003-kinect-the-technology-.html

3. Fernandez, D. "Installing and using the Kinect sensor (Beta 2 sdk), June 2011, Retrieved from: http://channel9.msdn.com/Series/Kinect SDKQuickstarts/ Understanding-Kinect-Hardware

4. Freedman Barak Binyamina, Shpunt Alexander, Machline Meir Ashdod, Arieli Yoel Jerusalem. *Depth mapping using projected patterns*. 2008. Retrieved April, 2012. www.freepatentsonline.com

5. Shotton J. (Microsoft Research Cambridge), Fitzgibbon A., Cook M., Blake A. *Real-time Human Pose Recognition in Parts from Single Depth Images*. ISBN: 978-1-4577-0394-2. 2011.

6. V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In Proc. CVPR, pages 2:775–781, 2005.

7. Mohammad, T. *Using ultrasonic and infrared sensors for distance measurement*, World academy of science, Engineering and technology 2009.

8. Garcia, A. *Physics of balance and weight shift*, 2011. Retrieved: May 2012 from http://www.algarcia.org/AnimationPhysics/BalanceTutorial.pdf

9. Pratap, R. and Ruina, A. *Introduction statics and dynamics*, Oxford university press, 2008, Retrieved from: http://ruina.tam.cornell.ed/BookCOMRuinaPratap.pdf

10. Drillis, R. Contini, R. and Bluestein, M. *Body segment parameters: A survey of measurement techniques*. 1964

11. Aldebaran Robotics. NAO key features. Retrieved April 2012. http://www.aldebaran-robotics.com/en/Discover-NAO/Key-Features/hardware-platform.html

12. Aldebaran Robotics. Who is NAO. Retrieved May 2012. http://www.aldebaran-robotics.com/en/Pressroom/About/NAO.html

www.manaraa.com

13. Aldebaran Robotics. *NAO software documentation 1.12.4*. Retrieved April 2012. http://developer.aldebaran-robotics.com/doc/1-12/

14. Kim, H. and Reed , K. "Gait modification in healthy individuals following robot-assisted balance training", Society for Neuroscience Annual General Meeting Abstracts, 2012, submitted.

15. Muscles of the Lower Extremity, Epidemiology and End Results (SEER) training modules, Retrieved in 20, May, 2012 from: http://training.seer.cancer.gov/anatomy/muscular/groups/lower.html

16. Gunnar Bolmsjö, Håkan Neveryd and Håkan Eftring. "Robotics in Rehabilitation". IEEE Trans. On Rehabilitation Engineering, 1995.

17. Izadi S, Kim D Hilliges O, Molyneaux D, Newcombe R, Kohli P, Shotton J, Hodges S, Freeman D, Davison A, and Fitzgibbon A. *KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera*. 2011. Retrieved from: http://research.microsoft.com/apps/pubs/default.aspx?id=155416

18. Shen S, Michael N, and Kumar V. *3D indoor exploration with a computationally constrained MAV*. University of Pennsylvania. Retrieved from: http://mrsl.grasp.upenn.edu/rss2011workshop/resources/Shen.pdf

19. Stone, E and Skubic, M. *Evaluation of an Inexpensive Depth Camera for Passive In-Home Fall Risk Assessment*. University of Missouri. 2011. Retrieved from: http://eldertech.missouri.edu/files/Papers/StoneE/Evaluation%20of%20an%20Inexpensive%20Depth%20Camera.pdf

20. TL;DR *Yet another anthro art tutorial*. Chapter 2: Human anatomy and figure drawing. Retrieved from: http://hippie.nu/~nocte/tutorial-currentchapter/xhtml-chunked/index.html.

www.manaraa.com

Appendices

www.manaraa.com

Appendix A: The Code of NAO and RABT

```
using  System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.Windows;
using System.Windows.Controls; using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;
using System.IO;
using System.Diagnostics;
using Aldebaran.Proxies;
using System.Timers;
using  Microsoft.WindowsMobile.DirectX;

namespace  new_kinect_2
{
    /// <summary>
    /// Interaction  logic  for  MainWindow.xaml
    /// </summary>
    public  partial  class  MainWindow  :  Window
    {
        public  MainWindow()
        {
            InitializeComponent();
        }

        KinectSensor  sensor1;
        TextWriter positions = new
        StreamWriter("joints.txt"); TextWriter center = new
        StreamWriter("center of mass.txt"); TextWriter
        oldpositions = new StreamWriter("old joints.txt");
        int TimeCounter = 1;

// Loading Kinect parameters
        private  void  Window_Loaded(object  sender,  RoutedEventArgs  e)
        {
            sensor1  =  KinectSensor.KinectSensors[0];

            var  parameters  =  new  TransformSmoothParameters
            {
                Smoothing  =  0.3f,
                Correction  =  0.0f,
                Prediction = 0.0f,
                JitterRadius = 1.0f,
                MaxDeviationRadius =
                0.5f
            };
            sensor1.SkeletonStream.Enable(parame
            ters);
            sensor1.DepthStream.Enable(DepthImageFormat.Resolution640x480
            Fps30);
            sensor1.ColorStream.Enable(ColorImageFormat.RgbResolution640x
            480Fps30); sensor1.AllFramesReady += new
            EventHandler<AllFramesReadyEventArgs>
    (sensor1_AllFramesReady);
```


www.manaraa.com

Appendix A (Continued)

```
    sensor1.Start();
           }
        void  sensor1_AllFramesReady  (object  sender,  AllFramesReadyEventArgs
        e)
         {
             image1.Source  =  e.OpenColorImageFrame().ToBitmapSource();
             skeletonjoints  (e);
         }

         double[] xx;
         double[]yy;
         double[] zz;

         // Loading NAO proxies
         MotionProxy motion = new MotionProxy("131.247.10.161", 9559);
    //naosim: 127.0.0.1 NAO: 131.247.13.118
         MemoryProxy memory = new
         MemoryProxy("131.247.10.161", 9559); List<float>
         oldspace = new List<float>();

         private  void  Pause_Click(object  sender,  RoutedEventArgs  e)
         {
             motion.setAngles("LShoulderPitch", 1.57f,
             0.2f); motion.setAngles("LShoulderRoll",
             0.0f, 0.2f);
             motion.setAngles("RShoulderPitch", 1.57f,
             0.2f); motion.setAngles("RShoulderRoll",
             0.0f, 0.2f);
             motion.setStiffnesses("Body", 0.0f);
         }
         private  void  Play_Click(object  sender,  RoutedEventArgs  e)
         {
             motion.setStiffnesses("Body",  0.5f);
         }

         double angleshoulderleft1
         = 0; double
         angleshoulderleft2 = 0;
         double angleshoulderright1
         = 0; double
         angleshoulderright2 = 0;
         double angleelbowleft = 0;
         double angleelbowright =
         0;

         private  void  skeletonjoints  (AllFramesReadyEventArgs  e)
         {
             canvas1.Children.Clear  ();
             SkeletonFrame  skeletonFrame  =  e.OpenSkeletonFrame  ();

             if  (skeletonFrame  !=  null)
             {
                 //  skeleton  data
                 Skeleton  []  skeletonData  =  new  Skeleton
                 [skeletonFrame.SkeletonArrayLength]
    ;
skeletonFrame.CopySkeletonDataTo(skeleton
```


www.manaraa.com

Appendix A (Continued)

```
                Data); oldspace =
                motion.getPosition("Torso", 0, true);

                foreach (var skeleton in skeletonData)
                {
                    var i = 0;
                    double
                    ground;
                    xx = new double[20];
                    yy = new double[20];
                    zz = new double[20];
                    double[] Cx;
                    Cx=new double[20];
                    double[] Cy;
                    Cy=new double[20];
                    double[] Cz;
                    Cz=new double[20];
                    double[] h;
                    h=new double[20];
                    double[] COMz;
                    COMz = new double[10];

                    if (skeleton.TrackingState==SkeletonTrackingState.Tracked)
                    {
                        foreach (Joint joint in skeleton.Joints)
                        {
                            DepthImagePoint depthpoint =
sensor1.MapSkeletonPointToDepth (joint.Position,                                ↙
sensor1.DepthStream.Format);
                            ColorImagePoint colorpoint =
sensor1.MapSkeletonPointToColor (joint.Position,                                ↙
sensor1.ColorStream.Format);
                            xx[i] = depthpoint.X;
                            yy[i] = depthpoint.Y;
                            zz[i] =
                            joint.Position.Z; h[i]
                            = 480 - depthpoint.Y;
                            canvas1.Children.Add(new Ellipse()
                            {
                                Margin = new Thickness(colorpoint.X,
                                colorpoint.Y, 0, 0), Fill = new
                                SolidColorBrush(Colors.Red),
                                Width = 10, Height =
                            10}); i = i + 1;
                        }

                        if (h[19] >=
                            h[15])
                            ground =
                            h[15];
                        else ground = h[19];

                        positions.Write(Math.Round(tim.Elapsed.TotalMilliseconds,
                        0) + " ")                                               ↙
;
                        for (i = 0; i < 20; i++)
                        {
```


www.manaraa.com

Appendix A (Continued)

```
                        Cz[i] = zz[2] -
                        zz[i]; Cx[i] = xx[i]
                        - xx[2]; h[i] = (h[i]
                        - ground)/2; Cy[i] =
                        (h[i] - h[2]);

                        positions.Write(+ xx[i] + " " + Math.Round(h[i], 0)
    + " " + Math.Round(zz[i], 2)+ " ");
                    }
                    TimeCounter =
                    TimeCounter+1; if
                    (TimeCounter % 10 == 0)
                    {
                        //NAO Leg movements
                        if  (h[15]  >  30)
                        {
                            motion.wbEnable(false);
                            motion.setAngles("RAnkleRoll", -0.33f,
                            0.1f); motion.setAngles("LAnkleRoll", -
                            0.33f, 0.1f);
                            motion.setAngles("RHipRoll", 0.33f,
                            0.1f); motion.setAngles("LHipRoll",
                            0.33f, 0.1f);
                            System.Threading.Thread.Sleep(1500);
                            motion.setAngles("LHipPitch", -1.25f,
                            0.1f); motion.setAngles("LKneePitch",
                            0.85f, 0.1f);
                        }
                        else  if  (h[19]  >  30)
                        {
                            motion.wbEnable(false);
                            motion.wbFootState("Fixed", "LLeg");
                            motion.wbEnableBalanceConstraint(true,
                            "Legs"); motion.setAngles("RAnkleRoll",
                            0.33f, 0.1f);
                            motion.setAngles("LAnkleRoll", 0.33f,
                            0.1f); motion.setAngles("RHipRoll", -
                            0.33f, 0.1f);
                            motion.setAngles("LHipRoll", -0.33f,
                            0.1f);
                            System.Threading.Thread.Sleep(1500);
                            motion.setAngles("RHipPitch", -1.25f,
                            0.1f); motion.setAngles("RKneePitch",
                            0.85f, 0.1f);
                            motion.wbGoToBalance("LLeg", 1.0f);
                        }
                        else  if  (h[15]  <  25  &&  h[15]  >  10  ||  h[19]
                        <  25  &&  h[19]  >  10)
                        {
motion.setAngles("RHipPitch", -0.436f, 0.1f); motion.setAngles("LHipPitch", -
0.436f, 0.1f); System.Threading.Thread.Sleep(1500);
motion.setAngles("RKneePitch", 0.7f, 0.1f); motion.setAngles("LKneePitch",
0.7f, 0.1f); motion.setAngles("LAnkleRoll", 0.0f,
```


www.manaraa.com

Appendix A (Continued)

```
                0.1f); motion.setAngles("RAnkleRoll",
                0.0f, 0.1f); motion.setAngles("RHipRoll",
                0.0f, 0.1f); motion.setAngles("LHipRoll",
                0.0f, 0.1f);
                motion.setAngles("LAnklePitch", -0.35f,
                0.1f); motion.setAngles("RAnklePitch", -
                0.35f, 0.1f); motion.wbGoToBalance("Legs",
                1.0f);
                System.Threading.Thread.Sleep(1500);
            }

            {
                // vectors for robot
                System.Windows.Media.Media3D.Vector3D
                shoulder_right1;
                System.Windows.Media.Media3D.Vector3D
                shoulder_right2;
                System.Windows.Media.Media3D.Vector3D
                shoulder_left1;
                System.Windows.Media.Media3D.Vector3D
                shoulder_left2;

                //vector of kinect
                System.Windows.Media.Media3D.Vector3D
                kinecthipcenter;
                System.Windows.Media.Media3D.Vector3D
                kinecthandleft;
                System.Windows.Media.Media3D.Vector3D
                kinecthandright;
                System.Windows.Media.Media3D.Vector3D
                kinectshouldercenter;
                System.Windows.Media.Media3D.Vector3D
                kinectshoulderleft;
                System.Windows.Media.Media3D.Vector3D
                kinectshoulderright;
                System.Windows.Media.Media3D.Vector3D
                kinectelbowleft;
                System.Windows.Media.Media3D.Vector3D
                kinectelbowright;

                kinecthipcenter = new
    System.Windows.Media.Media3D.Vector3D (xx[0], h[0], zz[0]);
    //kinecthipcenter.Normalize();
                kinecthandleft = new
    System.Windows.Media.Media3D.Vector3D (xx[7], h[7], (zz[7] * 100));
    //kinecthandleft.Normalize();
                kinecthandright = new
    System.Windows.Media.Media3D.Vector3D (xx[11], h[11], (zz[11] * 100));
    //kinecthandright.Normalize();
                kinectshoulderleft = new
    System.Windows.Media.Media3D. Vector3D(xx[4], h[4], (zz[4] * 100));
    //kinectshoulderleft.Normalize();
                kinectshoulderright = new
    System.Windows.Media.Media3D. Vector3D(xx[8], h[8], (zz[8] * 100));
    //kinectshoulderright.Normalize();
kinectshouldercenter = new System.Windows.Media.Media3D. Vector3D(xx[2], h[2],
zz[2]);
```


www.manaraa.com

Appendix A (Continued)

```
                        //kinectshouldercenter.Normalize();
                        kinectelbowleft = new
System.Windows.Media.Media3D.Vector3D (xx[5], h[5], (zz[5] * 100));
//kinectwristleft.Normalize();
                        kinectelbowright = new
System.Windows.Media.Media3D. Vector3D(xx[9], h[9], (zz[9] * 100));
//kinectwristright.Normalize();

                        shoulder_left1 = kinectshoulderleft -
                        kinectelbowleft; shoulder_left2 = kinectelbowleft
                        - kinecthandleft; shoulder_right1 =
                        kinectshoulderright - kinectelbowright;
                        shoulder_right2  =  kinectelbowright  -
                        kinecthandright;

                        //  new  axis
                        System.Windows.Media.Media3D.Vector3D newx = new
System. Windows.Media.Media3D.Vector3D(1, 0, 0);//new
System.Windows.Media.Media3D.Vector3D(xx [0], 0, 0);//kinectshoulderright
- kinectshoulderleft; //newx.Normalize();
                        System.Windows.Media.Media3D.Vector3D newy = new
System. Windows.Media.Media3D.Vector3D(0, 1, 0);//new
System.Windows.Media.Media3D.Vector3D(0, h [0], 0);//kinectshouldercenter -
kinecthipcenter; //newy.Normalize();
                        System.Windows.Media.Media3D.Vector3D newz = new
System. Windows.Media.Media3D.Vector3D(0, 0,
1);//System.Windows.Media.Media3D.Vector3D. CrossProduct(newx, newy);

                        angleshoulderleft1 =
System.Windows.Media.Media3D.Vector3D. AngleBetween(newx, shoulder_left1);
angleshoulderleft1 = 1.57 - (angleshoulderleft1 * Math.PI / 180);

                        angleshoulderleft2 =
System.Windows.Media.Media3D.Vector3D. AngleBetween(shoulder_left1, newz);
angleshoulderleft2 = angleshoulderleft2 * Math.PI / 180;

                        angleshoulderright1  =
                        System.Windows.Media.Media3D.Vector3D
.AngleBetween(newx, shoulder_right1); angleshoulderright1 = (1.57 -
 (angleshoulderright1 * Math.PI / 180));

                        angleshoulderright2  =
                        System.Windows.Media.Media3D.Vector3D
.AngleBetween(shoulder_right1, newz); angleshoulderright2 =
angleshoulderright2 * Math. PI / 180;

                        angleelbowleft =
System.Windows.Media.Media3D.Vector3D. AngleBetween(newz, shoulder_left2);
angleelbowleft = ((angleelbowleft * Math.PI / 180) -
 1.57);
                        angleelbowright =
System.Windows.Media.Media3D.Vector3D. AngleBetween(newz, shoulder_right2);
angleelbowright = (1.57 - (angleelbowright * Math. PI / 180));


                        if  (angleshoulderleft1  <  0.0f)  {
                        angleshoulderleft1  =  0.0f;
```


www.manaraa.com

Appendix A (Continued)

```
    }
                        if (angleshoulderleft1 > 1.57f) {
                        angleshoulderleft1 = 1.
57f; }
                        if (angleshoulderleft2 < 0.0f) {
                        angleshoulderleft2 = 0.0f;
    }
                        if (angleshoulderleft2 > 1.57f) {
                        angleshoulderleft2 = 1.
57f; }
                        if (angleelbowleft > 0) { angleelbowleft =
                        0.0f; }
                        if (angleelbowleft < -1.57) { angleelbowleft = -
                        1.57f; } motion.setAngles("LShoulderRoll",
                        (float)
(angleshoulderleft1), 0.1f);
                        motion.setAngles("LShoulderPitch", (float)
(angleshoulderleft2), 0.1f);
                        motion.setAngles("LElbowRoll",
                        (float)(angleelbowleft), 0.
1f);

                        if (angleshoulderright1 > -0.6f) {
                        angleshoulderright1 = -0
.68f; }
                        if (angleshoulderright1 < -1.4f) {
                        angleshoulderright1 = -1
.57f; }
                        if (angleshoulderright2 < 0.0f) {
                        angleshoulderright2 = 0.
0f; }
                        if (angleshoulderright2 > 1.17f) {
                        angleshoulderright2 = 1.
57f; }
                        if (angleelbowright < 0) { angleelbowleft =
                        0.0f; }
                        if (angleelbowright > 1.57) { angleelbowleft =
                        1.57f; } motion.setAngles("RShoulderRoll",
                        (float)
(angleshoulderright1 + 0.4), 0.1f);
                        motion.setAngles("RShoulderPitch", (float)
(angleshoulderright2), 0.1f);
                        motion.setAngles("RElbowRoll",
                        (float)(angleelbowright), 0.
1f);
                    }

                }

                Point COM1 = new Point(xx[0] + (xx[2] - xx[0]) *
                0.54, yy[0] + (yy
[2] - yy[0]) * 0.54);
                Point COM2 = new Point(xx[8] + (xx[9] - xx[8]) * 0.436,
yy[8] + (yy [9] - yy[8]) * 0.436);
                Point COM3 = new Point(xx[4] + (xx[5] - xx[4]) * 0.436,
yy[4] + (yy [5] - yy[4]) * 0.436);
Point COM4 = new Point(xx[9] + (xx[10] - xx[9]) *
```


www.manaraa.com

Appendix A (Continued)

```
                    0.682, yy[9] + (yy[10] - yy[9]) * 0.682);
                    Point COM5 = new Point(xx[5] + (xx[6] - xx[5]) * 0.682,
    yy[5] + (yy [6] - yy[5]) * 0.682);
                    Point COM6 = new Point(xx[16] + (xx[17] - xx[16]) * 0.433,
     yy[16] + (yy[17] - yy[16]) * 0.433);
                    Point COM7 = new Point(xx[12] + (xx[13] - xx[12]) * 0.433,
     yy[12] + (yy[13] - yy[12]) * 0.433);
                    Point COM8 = new Point(xx[17] + (xx[18] - xx[17]) * 0.433,
     yy[17] + (yy[18] - yy[17]) * 0.433);
                    Point COM9 = new Point(xx[13] + (xx[14] - xx[13]) * 0.433,
     yy[13] + (yy[14] - yy[13]) * 0.433);

                    COMz[0] = zz[2] + (zz[0] - zz[2]) * 0.54;
                    COMz[1] = zz[8] + (zz[9] - zz[8]) * 0.436;
                    COMz[2] = zz[4] + (zz[5] - zz[4]) * 0.436;
                    COMz[3] = zz[9] + (zz[10] - zz[9]) * 0.682;
                    COMz[4] = zz[5] + (zz[6] - zz[5]) * 0.682;
                    COMz[5] = zz[16] + (zz[17] - zz[16]) * 0.433;
                    COMz[6] = zz[12] + (zz[13] - zz[12]) * 0.567;
                    COMz[7] = zz[17] + (zz[18] - zz[17]) * 0.433;
                    COMz[8] = zz[13] + (zz[14] - zz[13]) * 0.433;
                    COMz[9] = ((COMz[0] * 0.532) + (COMz[1] * 0.031)
                    + (COMz[2] * 0.
    31) + (COMz[3] * 0.025) + (COMz[4] * 0.025)
                        + (COMz[5] * 0.115) + (COMz[6] * 0.115) +
                          (COMz[7] * 0.044) +
    (COMz[8] * 0.044)) / 0.962;

                    Point COM = new Point(((COM1.X * 0.532) + (COM2.X * 0.031)
    + (COM3. X * 0.031) + (COM4.X * 0.025) + (COM5.X * 0.025)
                        + (COM6.X * 0.115) + (COM7.X * 0.115) + (COM8.X *
    0.044) + (COM9.X * 0.044)) / 0.962,
                                    ((COM1.Y * 0.532) + (COM2.Y * 0.031)
                                    + (COM3.Y * 0.
    31) + (COM4.Y * 0.025) + (COM5.Y * 0.025)
                                    + (COM6.Y * 0.115) + (COM7.Y * 0.115)
                                      + (COM8.Y * 0
    .044) + (COM9.Y * 0.044)) / 0.962);

                    canvas1.Children.Add(new Ellipse()
                    {
                        Margin = new Thickness(COM.X, COM.Y,
                        0, 0), Fill = new
                        SolidColorBrush(Colors.Yellow), Width
                        = 10,

                        Height = 10
                    });

// Kinect with RABT
                    //Bar value:
                    progressBar1.Value = 50 - (COM.X -
                    xx[15]); progressBar2.Value = 50 -
                    (xx[19] - COM.X);
                    if ((50 - (COM.X - xx[15])) > 48 && (50 -
                    (xx[2] - xx[15])) > 48)
                    {
```


www.manaraa.com

Appendix A (Continued)

```
                    progressBar1.Foreground = new
                    SolidColorBrush(Colors.Green); label1.Foreground =
                    new SolidColorBrush(Colors.White);
                    label2.Foreground  =  new
                    SolidColorBrush(Colors.Green);
                }
                else
                {
                    progressBar1.Foreground = new
                    SolidColorBrush(Colors.Red); label1.Foreground =
                    new SolidColorBrush(Colors.Red); label2.Foreground
                    = new SolidColorBrush(Colors.White);
                }
                if  ((50  -  (xx[19]  -  COM.X))  >  48  &&  (50  -
                (xx[19]  -  xx[2]))  >  48)
                {
                    progressBar2.Foreground = new
                    SolidColorBrush(Colors.Green); label1.Foreground =
                    new SolidColorBrush(Colors.White); label2.Foreground
                    = new SolidColorBrush(Colors.Green);
                }
                else
                {
                    progressBar2.Foreground = new
                    SolidColorBrush(Colors.Red); label1.Foreground =
                    new SolidColorBrush(Colors.Red); label2.Foreground
                    = new SolidColorBrush(Colors.White);
                }
                center.WriteLine(+(Math.Round((COM.X), 2)) + " " +
(Math.Round((480 - COM.Y - ground) / 2, 0)) + " " +
                    (Math.Round(COMz[9],  2)));

            }
        }
    }
}

void  Kinectstop(KinectSensor  sensor1)
{
    if  (sensor1  !=  null)
    {
        sensor1.Stop();
    }
    //if  (sensor2  !=  null)
    //{
    //   sensor2.Stop();
    //}
}

private  void  UP_Click(object  sender,  RoutedEventArgs  e)
{
    if  (sensor1.ElevationAngle  <  sensor1.MaxElevationAngle)
    {
        sensor1.ElevationAngle  =  sensor1.ElevationAngle  +  5;
    }
}
```


www.manaraa.com

Appendix A (Continued)

```
        private void DOWN_Click(object sender, RoutedEventArgs e)
         {
             if (sensor1.ElevationAngle > sensor1.MinElevationAngle)
             {
                 sensor1.ElevationAngle = sensor1.ElevationAngle - 5;
             }
         }

        private void Window_Closed(object sender, EventArgs e)
        {
            Kinectstop(sensor1)
            ;
            //Kinectstop(sensor
            2);
            positions.Close();
            center.Close();
        }

        private void Close_Click(object sender, RoutedEventArgs e)
        {
            motion.setAngles("LShoulderPitch", 1.57f,
            0.2f); motion.setAngles("LShoulderRoll",
            0.0f, 0.2f);
            motion.setAngles("RShoulderPitch", 1.57f,
            0.2f); motion.setAngles("RShoulderRoll",
            0.0f, 0.2f);
            motion.setStiffnesses("Body", 0.0f);
            Kinectstop(sensor1);

            //Kinectstop(sensor
            2);
            positions.Close();
            oldpositions.Close(
            ); center.Close();
            Environment.Exit(0)
            ;
        }

    }
}
```

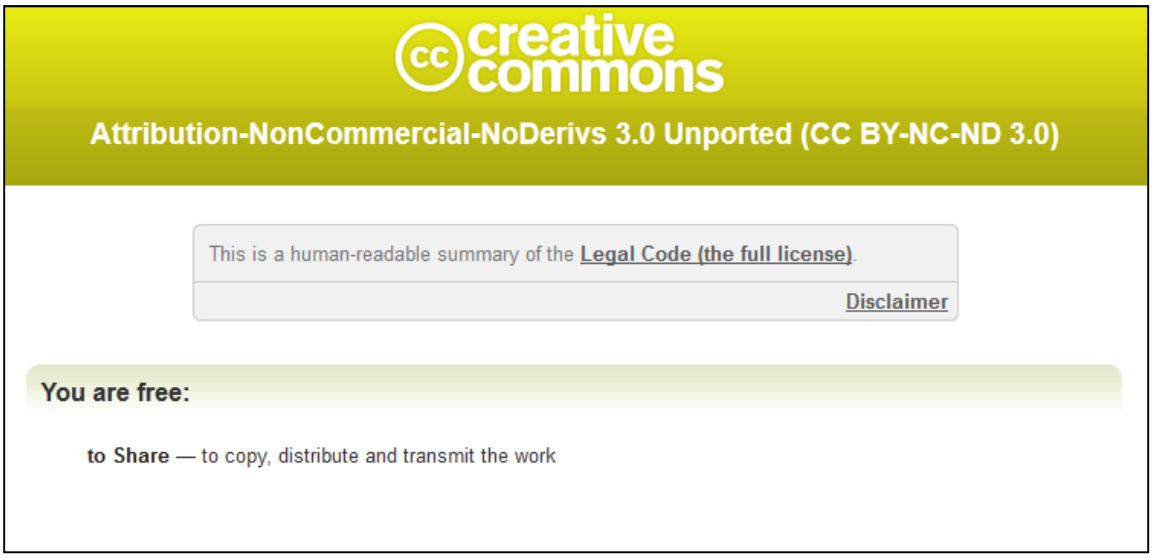

www.manaraa.com

Appendix B: Permissions

- The permission of using figures 2-6 and 2-7 is found in this website:

  http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en

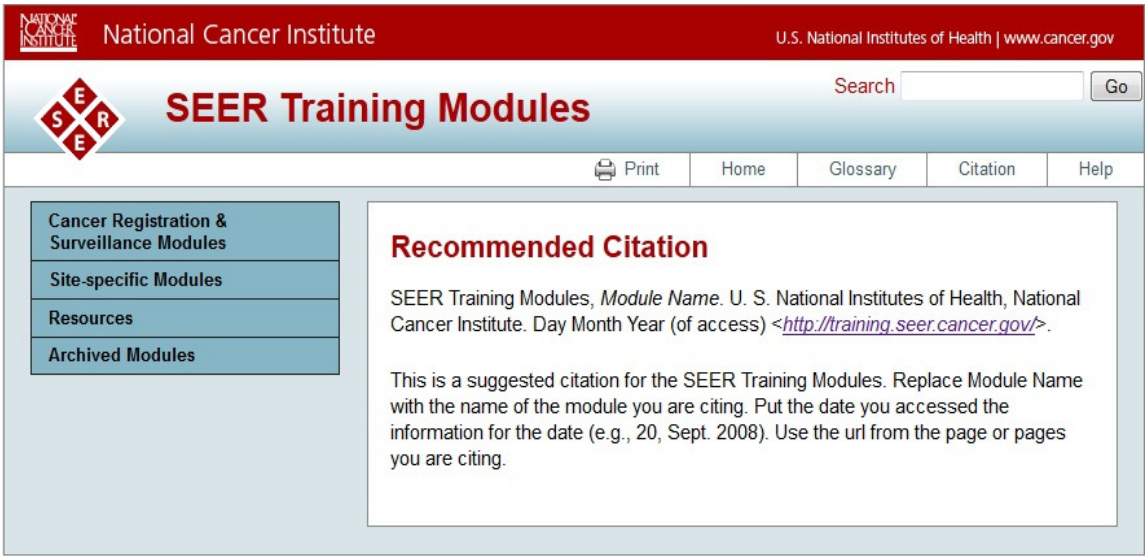
  Here is a snapshot of the web page:



- The permission of using figure 4-2 is found in this website:

  http://training.seer.cancer.gov/citation.html

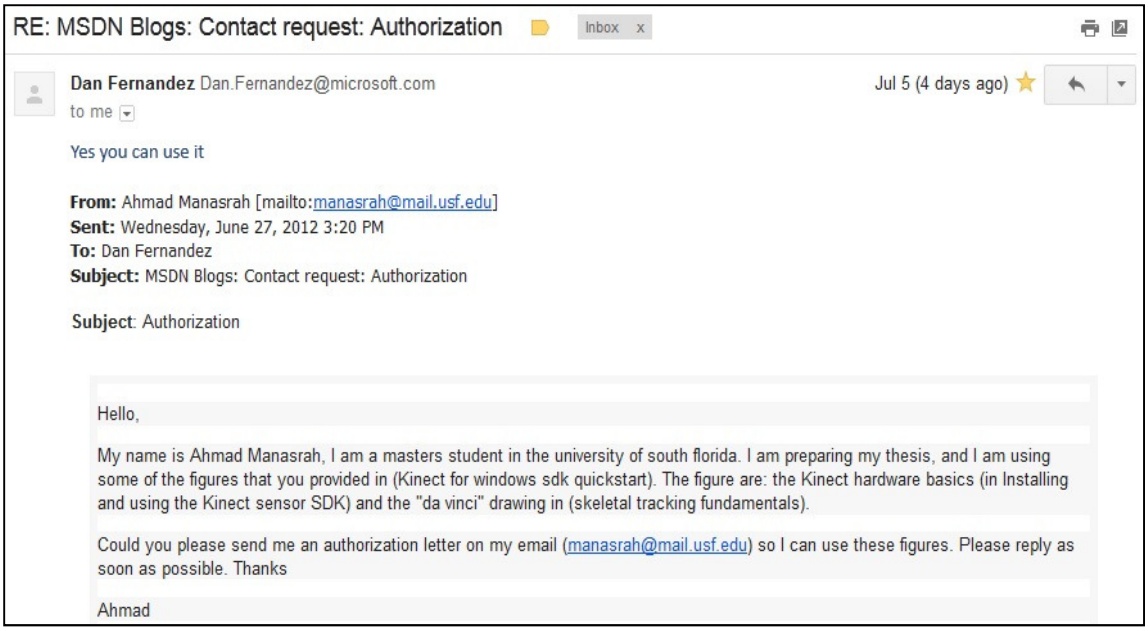
  Here is a snapshot of the web page:



www.manaraa.com

Appendix B (Continued)

- The permission of using figures 2-1, 2-2 and 2-10 are taken from the author Dan Fernandez. Here is a snapshot of the permission:



RE: MSDN Blogs: Contact request: Authorization

Inbox x

Dan Fernandez Dan.Fernandez@microsoft.com
to me

Jul 5 (4 days ago)

Yes you can use it

From: Ahmad Manasrah [mailto:manasrah@mail.usf.edu]
Sent: Wednesday, June 27, 2012 3:20 PM
To: Dan Fernandez
Subject: MSDN Blogs: Contact request: Authorization

Subject: Authorization

Hello,

My name is Ahmad Manasrah, I am a masters student in the university of south florida. I am preparing my thesis, and I am using some of the figures that you provided in (Kinect for windows sdk quickstart). The figure are: the Kinect hardware basics (in Installing and using the Kinect sensor SDK) and the "da vinci" drawing in (skeletal tracking fundamentals).

Could you please send me an authorization letter on my email (manasrah@mail.usf.edu) so I can use these figures. Please reply as soon as possible. Thanks

Ahmad

www.manaraa.com

Appendix B (Continued)

- The permission of using figures 2-4 and 2-5 are taken from the publisher IEEE. Here is a snapshot of the permission:

Copyright Clearance Center RightsLink®

Home | Create Account | Help

IEEE Requesting permission to reuse content from an IEEE publication

Title:	Real-time human pose recognition in parts from single depth images
Conference Proceedings:	Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on
Author:	Shotton, J.; Fitzgibbon, A.; Cook, M.; Sharp, T.; Finocchio, M.; Moore, R.; Kipman, A.; Blake, A.
Publisher:	IEEE
Date:	20-25 June 2011

Copyright © 2011, IEEE

User ID

Password

Enable Auto Login

LOGIN

Forgot Password/User ID?

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials.
Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

www.manaraa.com